2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)

Characterization and Prediction of Popular Projects on GitHub

Junxiao Han*, Shuiguang Deng*, Xin Xia †, Dongjing Wang‡, Jianwei Yin*
*Zhejiang University, Hangzhou, China
†Monash University, Australia
‡Hangzhou Dianzi University, Hangzhou, China
{junxiaohan, dengsg, tokyo1, zjuyjw}@zju.edu.cn, xin.xia@monash.edu

***Abstract*—GitHub is a large and popular open source project platform, which hosts various open source projects. Despite the prevalence of GitHub platform, not every project has gained high popularity. Identification of popular projects on GitHub can help developers choose proper projects to follow or contribute to, as well as provide guidance in building a popular project. In this paper, we propose an approach to predict the popularity of GitHub projects. We first conducted online surveys with GitHub users to determine the threshold (the number of stars of a project) of popular and unpopular projects. Next, we extract 35 features from both GitHub and Stack Overflow, which are divided into three dimensions: project, evolutionary, and project owner. A random forest classifier is built based on these features to identify popular GitHub projects. To evaluate the performance of our approach, we collect a large-scale dataset from GitHub which contains a total of 409,784 GitHub projects and 174,784 GitHub users. Our model achieves an average AUC of 0.76, which statistically significantly improves state-of-the-art by a substantial margin. We also study which features are of the most importance in distinguishing popular projects from unpopular ones. Experimental results show that number of branches, number of open issues, and number of contributors play the most important roles in identification of popular projects, and all of them have large effect size.**

***Index Terms*—GitHub project, Feature Engineering, Popularity, Prediction Model**

I. Introduction

GitHub[1] is a large and popular open source project platform, which hosts various open source projects in different domains. On GitHub, developers use fork and pull operations to create their own copies of a repository, and submit pull requests when they want project maintainers to merge their changes into main branch [2]. Also, if developers are interested in a project, they can "star" it. Prior studies use the number of stars as a proxy for the popularity of a project [3], i.e., the larger the number of star is, the more popular a project is.

In this paper, we aim to predict popular open source projects on GitHub based on 35 features mined from both GitHub and Stack Overflow. Prediction of popular projects on GitHub can bring a lot of benefits. For example, it can help developers to determine whether their projects will gain popularity and provide a guidance in how to gain more acceptance. Since different developers have different definitions of a popular project, we first conducted an online survey with 1,000 active developers on GitHub to assess which indicators (e.g., the number of stars, forks, or pulls) can be used as a proxy for the popularity of a GitHub project. As a result, we received 108 responses. 97 out of 108 participants agreed that the number of stars can be used as the proxy for project popularity. Then, the threshold for a popular project is chosed as 100, i.e., if a project has more than 100 stars, it can be regarded as a popular project.

Then, we extract 35 features from two information sources, i.e., GitHub and Stack Overflow, which are divided into three dimensions: project, evolutionary, and project owner. The project dimension refers to the project's innate features, e.g., the main programming language. The evolutionary dimension refers to the project's dynamic features, such as contributor count, commit count, the size of project increased since its creation. As for the owner dimension, we characterize the project owners' features. We mine the history of GitHub owners, such as repositories that they owned, list of developers that they followed, and list of followers. We also extract project owners' features from Stack Overflow, including reputation and the number of views. The rational is that we would like to investigate whether the project owners' behaviors on Stack Overflow will affect the popularity of projects that they owned on GitHub. Based on the 35 extracted features, we use random forest [4] as the default underlying classifier to build models, which can effectively determine whether a GitHub project is popular or not.

To our best knowledge, Borges et al.'s study [3] is the most related to ours. Borges et al. [3] performed an empirical study by analyzing the relationship between the various features and the popularity of GitHub projects. In this paper, to make a comparison, we leverage random forest to build a classifier based on their proposed features, and we refer to this baseline as RFB. Besides, we also make a comparison with other two baseline methods, i.e., Random Guess (RG) and Text Categorization(TC). Following prior studies in software analytics [5], [6], we use AUC as the evaluation metric. AUC measures the probability that a popular project is ranked higher than an unpopular project.

To evaluate the performance of our approach, we collect a large-scale dataset which contains a total of 409,784 GitHub projects and 174,784 GitHub users. In our study, we extract features from these projects in 2015, and use them to predict whether a project will be popular in 2018. Experimental results show that our approach achieves an average AUC of 0.76,

978-1-7281-2607-4/19/$31.00 ©2019 IEEE
DOI 10.1109/COMPSAC.2019.00013

which outperforms RG, RFB and TC by 52%, 43% and 49%, respectively. The results also show that among the 35 features extracted, number of branches, number of open issues, and number of contributors are the top three most important features in distinguishing popular projects from unpopular ones.

The main contributions of this paper are summarized as follows:

- We propose an approach which includes a total of 35 features to predict the popularity of GitHub projects.
- We evaluate the proposed approach on a large-scale dataset collected from GitHub, and experimental results show that the proposed approach statistically significantly outperform the baseline by a substantially margin. Besides, we find that the number of branches, the number of open issues, and the number of contributors are the top three most important features in determining whether a GitHub project is popular or not.

Paper Organization. The remainder of this paper is organized as follows. Section II describes the research questions that we explore in this study, elaborates the details of our data collection process and the experiment setup. Section III presents experimental results and answers to the research questions. Section IV discusses the threats to validity. Section V reviews prior studies related to ours. Finally, the conclusions and future works are given in Section VI.

II. Experiment Setup

In this section, we first introduce the collection of heterogeneous data sources such as GitHub dataset and Stack Overflow dataset, then we present the details of the 35 features used to predict popular projects, which are grouped into three dimensions: project, evolutionary, and owner. Then, we describe classifiers (i.e., random forest) used in this paper, evaluation metric, and experiment setup. In this paper, we would like to answer three research questions:

RQ1: Can we effectively predict the popularity of a GitHub project?

RQ2: How effectively does our model perform on different dimensions of features?

RQ3: Which features are of the most importance in distinguishing popular GitHub projects from unpopular ones?

A. Data Collection

GitHub Dataset: In this study, we collected GitHub projects and users via the GitHub API[1]. We crawled all the information of GitHub projects and users. Each project contains a project ID, full name, the ID of the owner who owned the project, the homepage, contributors, programming languages, the created time, the updated time and so on. Each user contains a user ID, name, email, the follower list, the following list, the ID of the repos that he owned, the created time and so on.

[1]https://api.github.com

All the projects collected spanning from October 2007 to December 2015. As a result, our dataset consisted of 28,362,019 projects and 15,647,255 users. In the preprocessing phase, we removed projects with less than two contributors because that many users use GitHub mainly for their own projects instead of collaborating with others [2]. After that, we analyzed the project age (i.e., the time period between the project creation date and December 31st, 2015) and removed the projects whose ages are less than 1 year, to reduce the bias caused by new projects. We also removed the projects which has only 0 or 1 star, since these projects have a high chance to be inactive [2]. As a result, we obtained 409,784 projects with 174,784 users, and 45,926 (11%) projects are popular. To make a prediction, we also obtained the stargazers of all the 409,784 projects in snapshot of April 2018 via the GHTorrent, i.e., we extracted features from projects on December 31st, 2015, and used them to predict whether a project will be popular in April 2018.

Stack Overflow Dataset: To figure out if project owners' behaviors on Stack Overflow have an influence on the popularity of GitHub project, we retrieved the identical users on GitHub and Stack OverFlow by comparing their email addresses, which is inspired by Vasilescu et al.'s study [7]. To reduce the bias due to the time difference in Stack Overflow and GitHub, we use the Stack Overflow data dump released in December 2015. After that, we found 16,679 common users with 54,034 GitHub projects and obtained the GitHub project owners' features on Stack Overflow, such as reputation and views, etc. Besides, some project owners might mention GitHub projects on Stack Overflow, and we also retrieved posts containing URLs that begin with *https://github.com*. By intersecting with previous intersection, we identified 8,121 users with 31,552 GitHub projects who have published posts with mentioning the GitHub projects on Stack Overflow.

B. Definition of Popular GitHub Projects

One challenge of this study is the definition of popular GitHub projects. In this paper, we performed a simple survey to developers on their perceptions of a popular project. In our survey, we asked three questions to developers: (1) What kinds of GitHub projects can be regarded as popular projects? (2) Can we use an indicator (e.g., the number of forks, downloads, or stars) as a proxy for the popularity? And (3) If you said yes to (2), what is the threshold between popular and unpopular GitHub projects?

We randomly chose 1,000 developers from our collected data, and sent emails to them. As a result, 97 out of 108 responses mentioned that the number of stars can be used as a proxy for the popularity of GitHub projects, and the median value of the threshold from responses is 100. As a result, we choose 100 as the threshold in this paper, i.e., if a project has more than 100 stars, we consider it as a popular project; otherwise it is regarded as an unpopular project.

TABLE I
STUDIED FEATURES.

Dimension	Feature Name	Description
Project	has_wiki	Whether the GitHub project has wiki, 1 if it has wiki, 0 otherwise
	has_pages	Whether the GitHub project has homepage, 1 if it has homepage, 0 otherwise
	readme_size	Calculated size of readme file for a given project
	branch_count	The number of branches for a given project
	main_programming _language	The main programming language for a given project
	topicVector [0,14]	The topic distribution of a project
Evolutionary	contributor_count	The number of contributors contributed to a given project
	commit_count	The number of commits for a given project
	increased_size	Lines of code increased since the creation date of the project
	open_issues	The number of open issues for a given project
	has_downloads	Whether the GitHub project has been downloaded
	has_issues	Whether the GitHub project has issues
Owner	owner_repos	The number of projects owned by the project owner
	owner_followers	The number of followers owned by the project owner
	owner_following	The number of developers that the project owner followed
	owner_gists	The number of gists owned by the project owner
	owner_organizations	The number of organizations the project owner belongs to
	owner_type	The type of project owner, such as general user, organization
	owner_so_reputation	The reputation of project owner on Stack Overflow
	owner_so_views	The number of views that the project owner received on Stack Overflow
	owner_so_mention	Whether the project owner mentions the project on Stack Overflow

C. Studied Features

Here, we carry out an exploration on the GitHub dataset to seek out the features that might play an important role in determining the popularity of GitHub projects. We extract 35 features as summarized in Table I, which are divided into three dimensions: project, evolutionary and owner. The project dimension is derived from the GitHub projects directly, where the evolutionary dimension is the evolutionary features of GitHub projects, and the owner dimension is composed of characteristics of project owner and the owner features from Stack Overflow.

Project Dimension refers to the innate features that are directly related to the GitHub projects. In this dimension, we focus on the projects themselves and extract 20 features from GitHub projects. Based on Borges et al.'s study, we take programming language and application domain into account. Then, the project dimension will be made up of 20 different features, which are *has_wiki*, *has_pages*, *readme_size*, *branch_count*, *main_programming_language* and *topic distributions*. The topic distributions are inferred by the LDA topic model. The details of project features can be seen in Table I.

Evolutionary Dimension: In this dimension, we assume that if a project has more contributors or commits and the size of it increases substantially since its creation, then the project has a higher chance to be popular. We name this dimension evolutionary dimension, since all of these features appear in the process of software evolution. Specially, we extract 6 features, i.e., *contributor_count*, *commit_count*, *increased_size*, *open_issues*, *has_downloads* and *has_issues*. The details of evolutionary features are shown in Table I.

Owner Dimension refers to features that are related with the GitHub project owners. In this dimension, we focus on the project owners and their features on both GitHub and Stack Overflow. As a result, we extract 9 features, which can be seen in Table I.

D. Classifiers

The 35 extracted features are then used to construct a prediction model to determine the popularity for a given GitHub project. In our study, we use random forest [4] as default classifier to construct the model. The main strengths of random forest is that it can automatically generate feature importance and is highly accurate in most cases. Besides, since random forest gathers up multiple results of decision trees to make a prediction, it exhibits robustness to noises and outliers. In this paper, we implement random forest on the basis of scikit-learn [8].

E. Evaluation Metrics

In this study, we use AUC to evaluate the effectiveness of the proposed model and the baseline methods. The AUC value reduces the ROC curve to a number, where higher AUC value means better performance, i.e., 0.5 corresponds to random guessing and 1 indicates a perfect prediction. Previous studies consider the AUC of 0.7 as promising performance [9] and AUC has been widely used in many software engineering researches, e.g. [6], [5]. In this study, we choose AUC as our performance measure for the following reasons: 1) AUC is a threshold independent measure [10]. 2) AUC is robust to class imbalance and is insensitive to class distribution [9]. 3) AUC has a statistical interpretation [9].

F. Experiment Setup

All the experiments are performed under the setting of 10 times stratified 10-fold cross-validation. In each round of 10-fold cross-validation, our dataset are randomly divided into 10 folds through the use of stratified random sampling. The purpose of stratified random sampling technique is to keep the class distribution of each fold the same as the original dataset. As a result, one of the 10 folds is chosen as the testing dataset, and the other nine folds are used as training dataset to train the classifier. We repeated the process 10 times, so that each of the 10 folds is used exactly once as the testing dataset. Then we calculate the average AUC scores across the 10 rounds of stratified 10-fold cross-validation. After that, 100 AUC scores are generated and then we can calculate the average AUC scores across the 100 AUC scores.

III. Results

In this section, we present answers to the three research questions proposed in Section II.

A. RQ1: Can we effectively predict the popularity of a GitHub project?

Approach: To answer this research question, we implement our proposed model using Random Forest classifier based on the 35 features extracted and adopt 10 times stratified 10-fold cross validation to estimate the accuracy of our model as default. To make a comparison, we adopt three baseline methods which are random guess (RG), random forest model built on Borges et al.'s proposed features (RFB) and text classification (TC), respectively. The baseline approaches are introduced as follows:

RG: RG is usually accepted as a baseline when no previous method exists to address the same research question [11]. This model randomly predicts whether a project is popular or not. As for the performance measures, the AUC of RG is 0.5 [11].

RFB: Borges et al. [3] analyzed the relationship between popularity and seven features, i.e., *project _age*, *commits*, *contributors*, *forks*, *main_programming_language*, *application domain* and *owner type*. In this paper, we compare our approach with the model built on Borges et al.'s proposed features. To avoid the potential bias, we remove the *forks* and *project age*. The number of forks and the number of stars are highly correlated, and both of them can be used as a proxy of popularity [12]. Borges et al. [3] also found that the project age has no correlation with the star of GitHub projects. To make a fair comparison, we build the baseline using random forest as the underlying classifier based on the remaining five features, and the baseline is referred to as RFB.

TC: As the last baseline, we use random forest to build a classifier based on the natural language description (i.e., readme file and project description) of GitHub projects.

Results: Results show that our approach achieves an average score of 0.76, which shows a large improvement compared with the baseline methods whose AUC scores are of 0.50, 0.53 and 0.51, respectively. We employ Wilcoxon signed-rank test [13] with a Bonferroni correction [14] to investigate the improvements of our approach over the baseline methods and determine whether the improvements are statistically significant. Besides, we compute Cliff's delta, which measures the effect size of the differences between two groups. The effect size is assessed using the thresholds provided by Cliff [15] [2].

As a result, we find that in terms of AUC, our approach on average improves RG, RFB, and TC by 52%, 43% and 49%, respectively. Statistical tests show that the improvements on AUC are all statistically significant and the effect sizes are large. Moreover, since RFB and TC only consider a subset of features to predict popular projects, the results also show that incorporating all features will significantly help to build an accurate model.

B. RQ2: How effectively does our model perform on different dimensions of features?

Approach: In this research question, we attempt to investigate how effectively does our model perform on different dimensions of our features. To this end, the 35 studied features are decomposed into three dimensions, which include project dimension, evolutionary dimension and owner dimension. This research question is designed to explore whether all features significantly help in our approach. With this intention, we make a comparison of the AUC scores of our approach (all dimensions) and the random forest models using features in each dimension.

We learn the random forest models based on features in each dimension. As a result, we construct three random forest models for each dataset and the constructed random forest models are denoted the same as the dimension names (i.e., project, evolutionary and owner, respectively). After that, we experiment the three random forest models in each dataset and obtain the average AUC scores in the 10-times 10-fold cross-validation.

Results: Results show that our approach (all dimensions) achieves substantial improvements over the prediction models built on project, evolutionary and owner dimensions whose AUC scores are of 0.58, 0.66 and 0.53, respectively. Furthermore, the improvements are statistically significant and the effect sizes are large, which implies that incorporating all features will significantly help to build an effective model compared to three models built on different features in different dimensions. Moreover, the evolutionary dimension plays a more crucial role in determining the popularity of GitHub projects than the other two dimensions, and the owner dimension does not perform as well as the other two dimensions.

C. RQ3: Which features are of the most importance in distinguishing popular GitHub projects from unpopular ones?

Approach & Results: In this research question, we aim to identify the most important features to distinguish popular from unpopular projects. Being aware of what features impact

[2]Cliff defines a delta of less than 0.147, between 0.147 and 0.33, between 0.33 and 0.474 and above 0.474 as negligible, small, medium, large effect size, respectively.

TABLE II
IMPORTANCE OF THE FEATURES OF THE GITHUB PROJECTS AS RANKED ACCORDING TO THE SCOTT-KNOTT ESD TEST. THE SECOND AND THIRD COLUMNS SHOW P-VALUES, CLIFF'S DELTA FOR THE FEATURES. THE FEATURES WITH NON-NEGLIGIBLE EFFECT SIZES ARE IN BOLD.

Groups	Features	P-value	Cliff's delta
1	**branch_count**	**<0.001**	**0.64(Large)**
2	**open_issues**	**<0.001**	**0.63(Large)**
3	**has_issues**	**<0.001**	**0.21(Small)**
4	**contributor_count**	**<0.001**	**0.44(Medium)**
5	**readme_size**	**<0.001**	**0.30(Small)**
6	**commit_count**	**<0.001**	**0.33(Small)**
7	**increased_size**	**<0.001**	**0.21(Small)**
8	main_programming_language	<0.001	0.06
9	owner_repos	<0.001	0.02
10	topicVector.14	>0.05	-0.12

popular projects the most can help to gain a deeper understanding on how to formulate a popular project. The detailed process is as follows:

Step 1: Correlation Analysis. Feature selection aims to remove correlated features which may lead to poor models [16]. To reduce the bias of the model and avoid the interaction between features, we make the correlation analysis and remove 2 features in the dataset, which are *owner_followers* and *owner_so_views*. As a result, 33 features are remained.

Step 2: Redundancy Analysis. After correlation analysis, we reduce the collinearity among the features and the **redun** function provided by the R package **rms** is applied. By redundancy analysis, we find that none of the remaining features are redundant.

Step 3: Important Features Identification. For that random forest allows for the estimation of feature importance, we build a new random forest model based on the 33 remaining features using the python tool of **sklearn** [8]. In the training process, we use the **feature_importances_** function in the python package **RandomForestClassifier** to compute the importance of features. In this process, 10 times 10-fold cross-validation is used. To determine which features are of the most importance for the whole dataset, we employ Scott-Knott Effect Size Difference (ESD) test [17] using the importance values from all 10 rounds of 10-fold cross-validation.

Table II presents the top 10 importance ranking of the features according to the Scott-Knott ESD test. The results reveal that *branch_count*, *open_issues* and *has_issues* are ranked in the top three important features that affect the random forest model to predict the popularity of GitHub projects. Moreover, two of them belong to the evolutionary dimension, which is consistent to the conclusions in RQ2 that evolutionary dimension plays a more important role in determining the popularity of GitHub projects than project dimension and owner dimension.

Step 4: Effect of Important Features. To understand the impact of each feature, we compare the values of the remaining features between popular and unpopular GitHub projects. We apply the Wilcoxon rank-sum test [18] with Bonferroni correction to analyze the statistical significance of the difference between popular and unpopular GitHub projects. We use Cliff's delta to measure the effect size of differences between the two groups of GitHub projects.

Table II presents the top 10 p-values and Cliff's delta of the features. As we can see, *branch_count*, *open_issues* and *contributor_count* are the most important features to distinguish popular from unpopular GitHub projects, while *branch_count*, *open_issues* and *has_issues* are more important in affecting the random forest model. These features are ranked in the top four most important features, and they have statistically significant and non-negligible positive effect.

IV. THREATS TO VALIDITY

One threat to internal validation relates to the fairness of our results derived from online survey. Because the users are random selected according to their commits, which may make the results derived from survey not be accurate enough. The other threat relates to the stars of the GitHub projects. In this paper, we use the number of stars of a project as a proxy for popularity, which may not be perfect enough. In the future, we plan to reduce this threat by using compound indicators (e.g., the number of forks and the number of stars) as the proxy of popularity. Threats to external validity are related to the features. Except the features considered in this paper, there might be additional features that could be more relevant to the popularity of GitHub projects. Future studies should consider more features to make the relationship between features and the popularity of GitHub projects more precise.

V. RELATED WORK

In this section, we first describe the related work on popularity of GitHub projects. Then, we describe other studies on GitHub.

A. Studies on Popularity of GitHub Projects

To the best of our knowledge, Borges et al.'s study [3] is the most related to ours. Borges et al. [3] performed an empirical study on the popularity of projects hosted on GitHub. They analyzed 7 features on project popularity and identified four main patterns of popularity growth, and all of the analyses were based on 2,279 popular GitHub repositories.

Compared with Borges et al.'s study, our study aims to address a related but different problem. Specifically, we propose an automated approach to predict the popularity of a project. Our study uses a larger scale dataset, which is composed of 409,784 GitHub projects, while Borges et al.'s dataset only consists of 2,279 popular GitHub projects. Moreover, our approach considers more features. We extract not only features mentioned in Borges et al.'s study, but also more domain-specific features from both GitHub and Stack Overflow. Then, we build a model to predict the popularity of GitHub projects, but Borges et al.'s study mainly focused on the empirical analysis of the correlations between the features and the popularity of GitHub projects. Last but not least, their study only analyzed features on the popular GitHub projects, but did not investigate the unpopular ones.

B. Studies on GitHub

GitHub [1] has gained more and more popularity and attracted many researchers. The most comprehensive studies on GitHub were achieved by Kalliamvakou et al. [2] and Gousios et al. [19]. In these studies, the authors described the quality and properties of the available GitHub dataset, which can help researchers better understand the characteristics of the projects and users on GitHub. Ray et al. [20] performed an analysis and statistics on GitHub dataset, and attempted to explore the effect of programming language features on software quality. According to the results, language design does have a significant, but modest effect on software quality. Bissyand et al. [21] also made an analysis of the popularity, interoperability, and impact of various programming languages. In this paper, we also take programming language into account.

Due to the prevalence of GitHub, many researches are proposed to evaluate contributions on GitHub [22], [23]. Tsay et al. [22] focused on evaluating pull requests to study the contribution on GitHub and also analyzed the association of various technical and social measures with the likelihood of contribution acceptance. Gousios et al. [23] provided insights into the factors that they considered in their decision-making process to accept or reject a contribution.

VI. Conclusion

In this paper, we propose an automated approach to predict popular projects on GitHub. We extract 35 features to characterize the GitHub projects and these features are grouped into three dimensions: project, evolutionary and owner. To explore the effectiveness of our approach, we conduct an experiment on GitHub dataset with a total of 409,784 projects. Our experimental results show that our approach can achieve an average AUC of 0.76, which outperforms the baseline methods of RG, RFB, and TC by 52%, 43% and 49%, respectively. We also find that *branch_count*, *open_issues*, and *contributor_count* are the most important features to predict the popularity of GitHub projects, and all of them have large effect size. In the future, we intend to extract more features that can have a significant influence on the popularity of GitHub projects, and design a better approach to improve the performance further.

Acknowledgment

This research was partially supported by the National Key Research and Development Program of China (No. 2017YFB1400601), Key Research and Development Project of Zhejiang Province (No. 2017C01015), National Science Foundation of China (No. 61772461), Natural Science Foundation of Zhejiang Province (No. LR18F020003 and No.LY17F020014).

References

[1] L. Yu, A. Mishra, and D. Mishra, "An empirical study of the dynamics of github repository and its impact on distributed software development," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2014, pp. 457–466.

[2] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, 2016.

[3] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. IEEE, 2016, pp. 334–344.

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

[5] Y. Fan, X. Xia, D. Lo, and S. Li, "Early prediction of merged code changes to prioritize reviewing tasks," *Empirical Software Engineering*, pp. 1–48, 2018.

[6] S. Wang, T.-H. Chen, and A. E. Hassan, "Understanding the factors for fast answers in technical q&a websites," *Empirical Software Engineering*, pp. 1–42, 2017.

[7] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Social computing (SocialCom), 2013 international conference on*. IEEE, 2013, pp. 188–195.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[9] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[10] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[11] X. Xia, E. Shihab, Y. Kamei, D. Lo, and X. Wang, "Predicting crashing releases of mobile applications," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2016, p. 29.

[12] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in github," *Empirical Software Engineering*, vol. 22, no. 1, pp. 547–578, 2017.

[13] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[14] H. Abdi, "Bonferroni and šidák corrections for multiple comparisons," *Encyclopedia of measurement and statistics*, vol. 3, pp. 103–107, 2007.

[15] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological Bulletin*, vol. 114, no. 3, p. 494, 1993.

[16] A. Mockus, M. Nagappan, and A. Hassan, "Best practices and pitfalls for statistical analysis of se data," in *Proc. ICSE*, 2014.

[17] H. Li, W. Shang, Y. Zou, and A. E. Hassan, "Towards just-in-time suggestions for log changes," *Empirical Software Engineering*, pp. 1–35, 2016.

[18] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[19] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[20] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 155–165.

[21] T. F. Bissyandé, F. Thung, D. Lo, L. Jiang, and L. Réveillere, "Popularity, interoperability, and impact of programming languages in 100,000 open source projects," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. IEEE, 2013, pp. 303–312.

[22] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th international conference on Software engineering*. ACM, 2014, pp. 356–366.

[23] G. Gousios, A. Zaidman, M.-A. Storey, and A. Van Deursen, "Work practices and challenges in pull-based development: the integrator's perspective," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015, pp. 358–368.