Preliminary study: Exploring GitHub repository metrics

Guzel Safiullina
Innopolis University
Innopolis, Russia
gu.safiullina@innopolis.university

Aidar Gumerov
Innopolis University
Innopolis, Russia
a.gumerov@innopolis.university

Gcinizwe Dlamini
Innopolis University
Innopolis, Russia
g.dlamini@innopolis.university

Giancarlo Succi
Innopolis University
Innopolis, Russia
g.succi@innopolis.ru

***Abstract*—GitHub is the largest platform for code storage and development and is currently the source of data on software projects. It is important to understand how repository properties affect code quality, project popularity, and find dependencies between different repository metrics. In this paper, a qualitative and quantitative analysis of more than 700 repositories and 81 metrics was conducted. Descriptive statistics, statistical tests, and correlation analysis were investigated. An analysis of the resulting descriptive statistics was conducted. The correlation analysis highlighted strongly correlated metrics and provided a theoretical justification for the dependencies obtained. In addition, clustering of repositories was performed and discussion of obtained groups of repositories is presented.**

***Index Terms*—GitHub, open-source, clustering, correlation analysis, statistics**

I. Introduction

There are now many teams working on open source software development. Software projects differ in scale, where one person or a large team can work on one project. Projects can also be either commercial or social. GitHub is the largest portal where developers of many projects interact. In addition GitHub serves as a knowledge and code base which for years has been a source of success for IT companies starting from start-ups to giant companies [27]. GitHub has tens of millions of registered users, and the number of repositories exceeds two hundred million [10].

Harnessing the knowledge GitHub presents is crucial for IT businesses for reasons such determining project maintainability, adoption of software development practices used by successful open source projects [7], [13], [18]. Open source projects are an effective way to develop software. Therefore, it is important to understand how the metrics that characterize a repository affect the quality of the code, its popularity, and what insights can be gained from this data [4], [31].

For years researchers have developed and proposed approaches for analysing open source projects to formulate strategies that propel software projects to success and high quality [35] [31].To assess code quality, a defect density calculation is used, which can be predicted based on repository metrics such as number of authors, number of downloads, and so on [22], [29].

Over the recent years researchers have proposed various approaches on mining knowledge from GitHub repositories metrics [32], [33]. However understanding the essence and the relationship between the metrics has remained a complex task and open area of research. Statistical and mathematical modelling approaches which serves as bases for most machine learning approaches for learning complex underlying patterns of data have been proposed and there is still a room for improvement [21]. For example clustering of software projects using selected metrics reveals different insights since the metrics changes with time.

This paper contains a qualitative and quantitative study of GitHub repository metrics. The main purpose of the paper is to establish relationship between the metrics of the project, to study the distribution of the data and get possible insights from the data. Secondly, our goal is to cluster repositories by their scale and purpose. For example, a learning project is likely to contain a much smaller number of commits and participants compared to commercial projects of large companies. Our approach is mainly based on fundamentals of statistical methods, correlation analysis and clustering.

The remainder of this paper is organized as follows. The related work is presented in section II. Section III provides the methodology which are the methods used in the project. Section IV presents the results section presents overview of extracted data, while section V discusses the results. Lastly section VI presents the conclusion and future directions.

II. Related Work

The challenge of analysing and extracting knowledge from software project metrics has has existed for a long time and the software engineering research community has proposed both statistical and machine learning models to address the challenge [15], [19]. As GitHub has grown drastically over the years, the challenge has escalated and now comes under the scope of Big data.

Kalliamvakou et al. [14] in their study conducted a quantitative and qualitative analysis aimed at understanding the characteristics of the repositories in GitHub and how users take advantage of GitHub's main features. As a result the authors concluded that the vast majority of repositories are individual projects and Github over ther years has become one of valuable source for knowledge in the domain of sotware development. Analyzing Github repository statistics is an important tool for many different application tasks. One of these tasks is to predict the popularity of the project.

Borges et al. [5] proposed an approach to predict popularity of repositories. The authors used data on the 5000 most popular repositories, cluster it and predict the number of project stars in each cluster. With similar objective Ren et al. [24] proposed an approach for predicting the popularity of repositories using data on the behavior of stargazers. A study of the statistics of more than 2,000 repositories mentioned in scientific publications has shown that the number of stars and forks is distributed according to a power law [9]. However github stars are not a sufficient indicator of a software health.

Another type of task is clustering repositories on the GitHub. In this paper, the authors performed a hierarchical clustering of GitHub repositories based on keywords [34]. Also, the study of repository metrics can be useful in studying trends in a particular area. For example, the authors of this article collected information about COVID19 repositories and analyzed their characteristics [30]. They have analyzed the number of repositories, the time-dependence of the number of repositories, the topic, and the characteristics of development.

In a study conducted by [6], the researchers studied the effect of test driven development in GitHub. The effect of this approach on such metrics as commit velocity and number of bug-fixing commits was studied. Repository statistics (forks and commits) can be used to cluster repositories and identify vulnerable groups [16]. Putting different metrics together and gaining insights to improve software development process or product still remains an area of research.

In light of the aforementioned proposed approaches, we propose a statistical approach and machine learning based approach to analyse GitHub repositories and extract knowledge which could be useful in improving the quality of software product of development process.

III. Methodology

Our proposed approach to analyse GitHub repositories is presented in Fig. 1. It contains five main stages. The following subsections outlines the details about each stage.

A. Data extraction and preprocessing

Dataset includes information about public Github repositories. The data used in this paper is retrieved through GitHub Repository Statistics API. Dataframe contains 83 metrics. The study includes a variety of information about the dataset. A detailed description is provided in the list: commits, contributors, forks, issues, repo, pulls, releases, stars, workflow runs. As part of Data preprocessing we remove duplicates and Nan-values.

B. Exploratory data analysis

We will visualize data using dimensionality reduction.

1) Dimensionality reduction: To find the most informative features. Principal Component Analysis (PCA) [1] will be applied. PCA allows to extract most important information and this method is useful for data visualization.

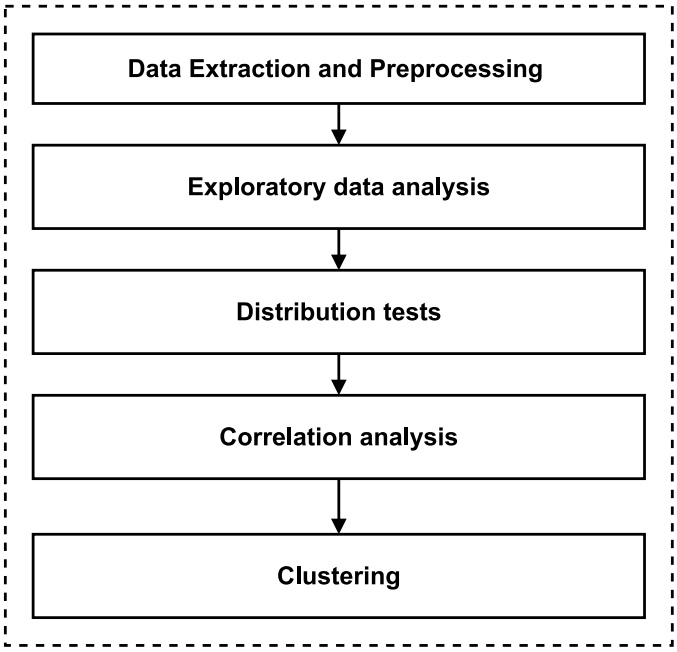

Fig. 1. Our proposed approach

C. Simple statistics

To define distribution properties the following statistics will be calculated: mean, standart deviation, median, mode, 25%- and 75% quantile [11].

D. Distribution test

1) Normality test: The Kolmogorov Smirnov test is a statistical nonparametric test that is applied to continuous univariate distributions, and which determines whether a given sample belongs to a particular distribution [23]. The Kolmogorov Smirnov test is also used to check if two samples belong to the same distribution. The advantage of this test is that different distribution laws can be chosen as a reference distribution, including normal distribution, Poisson distribution, exponential distribution, and others [26]. Test data for normality with Kolmogorov-Smirnov test: H_0 the data is normally distributed, H_α - data is not normal.

E. Confidence interval

Confidence interval is a range by which you can estimate an unknown value. The confidence interval is calculated at a certain level of confidence. The most commonly used confidence level is 95%, but other values can also be used [8]. In research, confidence intervals allow us to estimate the likely range of values of a parameter for the entire population. Confidence intervals, in turn, allow you to estimate the true average value of the general population [12]. Using the bootstrapping technology, we generate 100 datasets (randomly select repositories from the initial retrieved data). Let's define the size such samples as 10% of the main population. Now, for each such sample, we calculate the statistics that were calculated for the population. After we calculate 95% confidence interval.

F. Correlation analysis

To define linear and non-linear dependencies between features correlational matrices will be calculated [11].

1) Pearson correlation: Pearson correlation to find linear dependencies in data [3]

$$r = \frac{cov(X, Y)}{\sqrt{Var(X)Var(Y)}} \quad (1)$$

2) Spearsman correlation: Spearsman Correlation. Data should be normally distributed . [20]

$$r_P = \frac{cov(R_X, R_Y)}{\sqrt{Var(R_X)Var(R_Y)}} \quad (2)$$

where is R_{X_i} - is rank of this observation

G. Clustering

We can distinct several groups of projects: study projects, empty projects, large-scale business projects, etc. To formalize this we can apply different clustering methods. First and simplest one is k-means algorithm [17].

1) K-means algorithm: The k-means algorithm belongs to the group of unsupervised learning algorithms. This algorithm allows you to allocate different groups for data that does not have class labels. The algorithm inputs the number of clusters and then k-means minimizes the function of the sum of squares of distances from a point to the center of the cluster. The disadvantage of the algorithm is that you need to know the number of clusters in advance. Therefore, the number of clusters can be determined using the Elbow method.

2) DBSCAN: DBSCAN - Density-Based Spatial Clustering of Applications with Noise is another clustering algorithm that works with unlabeled data [25]. The algorithm is based on clustering based on the criterion of density of located points. This algorithm is effective when working with data that are homogeneous in density. The principle of the algorithm is as follows: the algorithm finds clusters with high density, with clusters separated by areas with low density. As a result, the formed clusters can take any shape, in contrast to k-means, where clusters must have a convex shape. The DBSCAN algorithm also allows the detection of noisy points that do not belong to any cluster [2]. In this case, real data sets contain clusters of different density with boundaries of different degrees of fuzziness. In conditions where the density of some boundaries between clusters is greater than or equal to the density of some isolated clusters, we have to sacrifice something in the efficiency of the algorithm.

IV. Results

A. Data extraction and preprocessing

We have collected data 764 repositories. Duplicates were removed. Further, where the missing values occupy less than 30%, we fill in based on the neighboring values - (take the average or zero, for example, sometimes empty commit cells do not mean the absence of information, but the fact that there were no commits). Also we exclude 2 metrics ('commits avg per day') which contains only zero values. Total number of repositories left after preprocessing was 732 and number of metrics was 81.

B. Exploratory Data analysis

We have built histograms for each metric, however, not all of them are informative. Also we applied PCA and have found that 20 features contains 90 % of information. We visualized the data after dimensionality reduction using PCA.

C. Calculating simple statistics and confidence intervals

Mean, median, mode, minimum, maximum, standart deviation were calculated. Thus, we determine the quantiles of t-statistics for the population and so on for 100 samples in order to simply assess how close these intervals are. Apply bootstraping to estimate means. Bootstrapping was applied and mean for each feature was estimated by 25% and 75% quantiles. For all metrics, the standard deviation is much larger than the mean. For all metrics median is smaller than mean. This fact is due to skewness of data to zero. For 66 out of 81 metrics mode is equal to zero, for 14 of them mode is equal to 1 or 2 and for "repo_age_days" mode equals to 482. Our data contain a large number of observations for which the metrics are close to zero, but several observations are large projects for which these metrics are much larger than zero. This creates a situation in which the mean, mode, and median are close to or equal to 0, and the standard deviation is very large. All of this suggests that the data are not normally distributed.

D. Test data for normality

After applying Kolmogorov-Smirnov test we found out that data us not normally distributed, as a result we can not assume that Pearson correlation coefficient does not belong to T-distribution.

E. Correlational analysis

To find linear relationship Pearson correlation coefficients r was first calculated. After we have calculated Spearsman correlation matrix. We choose 22 pairs with correlation coefficient (r_s) bigger than threshold ($r > 0.8$) and explain their dependence.

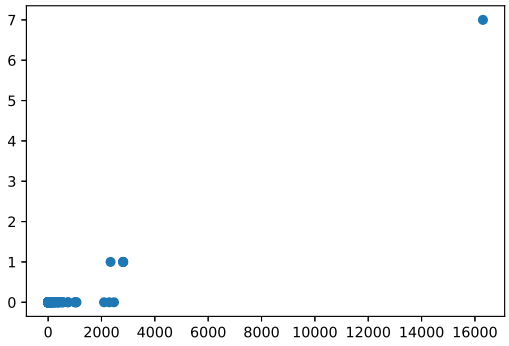

Fig. 2. Forks count vs forks average per day $r_{xy} = 0.96, r_{Spearman} = 0.1$

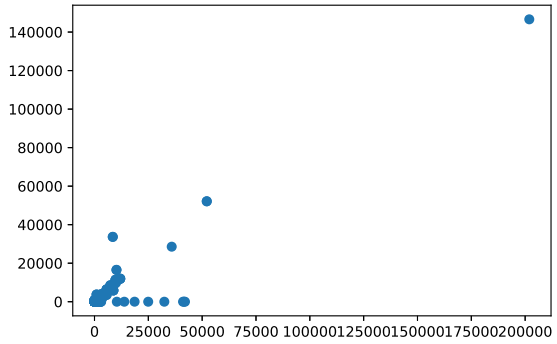

Fig. 3. Commits average versus contributors top

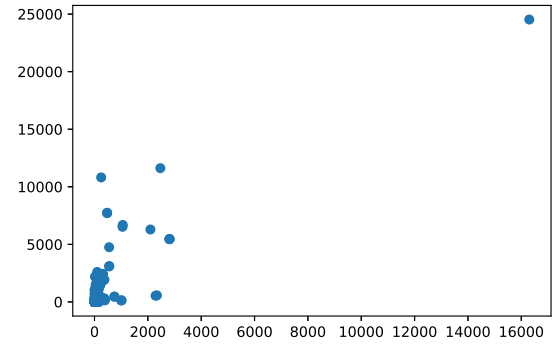

Fig. 4. Releases average versus stars max per day

After calculating the variance it was found that 40 variables explain 90% of all information.

This is due to the fact that we have 20 metrics highly correlated with each other in this way - they are linearly related and can be expressed through each other.

V. Discussion

This section presents the discussion of the obtained results based on our proposed approach in section III

A. Correlational analysis

We have found pairs for which Spearsman correlation coefficient is significantly distinct from Pearson correlation coefficient ($|r_s - r| > \varepsilon$). Let's look at the dependence of these data on each other where the difference is very large: $\varepsilon > 0.4$. When analyzing such data, we saw that at least one of these two metrics within a pair contains 90% zeros. Then the small Spearman coefficient is explained by a very large randomness near zero.

To estimate correlational coefficients bootstrapping was applied. First of all, we decided on the size of the samples (20) and their number. After generating these datasets, we calculated how many of them fit the null hypothesis and got 70% of the total. After that we will analyze the remaining 30% and what is wrong with them, because it is possible that outliers are contained in them. In the following graph, we have shown a histogram for one pair of highly correlated metrics to see how many samples contain strange data.

As far as can be estimated, very few of the "strange" samples are located near the 0.8 trash hold. In this case, many coefficients lying near zero can be considered as an anomaly. for such anomalies, we also calculate the statistics and compare them with their counterparts from the population.

On the graphs, we see that for all such pairs there is one special point - outlier. So you need to check how the data behaves without it. We also recalculated the Pearson coefficient without outliers and predicted that it for features began to fall for some pairs of metrics by 40-50%. Thus, a linear dependence is no longer necessary, because only one point out of 700 corrects the dependencies so significantly. This observation is project is helm/charts[1]. It is a package manager for Kubernetes. This project has more than 17000 forks and more than 3400 contributors.

As a result, we also analyzed several pairs of metrics that have the highest correlation coefficient. Their list is presented below. Pairs with the highest Spearman correlation coefficient (correlation coefficient higher than 0.8):

- Commits days since first - Repository days.
  The longer the life of the project, the longer time from first commit
- Commits average per day - Commits maximum per day.
  Since the average of a set depends on the maximum element of this set linearly, it is obvious that they are strongly correlated
- Contributors top average commits - Contributors top average participation weeks
  Also assume a linear dependence - the more contributors per week, the more commits they will make.
- Forks count - Repository network members.
  When the command is expanded, the number of forks for parallelizing the process increases.
- Issues total comments - Issues count.
  The more questions, the more commentators to them.
- Repository workflows - Workflow runs count.
  It is almost same.
- Repository watchers - Stars count.
  The more people who follow the project, the more positive feedback (stars) the project will receive.
- Pulls total lines added - Pulls total lines removed.
  As the size of the project grows and the number of lines of code increases, the possible number of errors and the number of deleted lines of code grows accordingly.
- Releases count - Releases tags.
  The metric data represent practically the same thing and therefore have a correlation coefficient of 1.
- Workflow average duration - Workflow average failure duration.

[1]https://github.com/helm/charts

The work process includes failure - when the set itself grows uniformly, then the subset of this set also grows and vice versa

B. Clustering

1) K-means: Visualization on the two main components is presented on Figure 5.

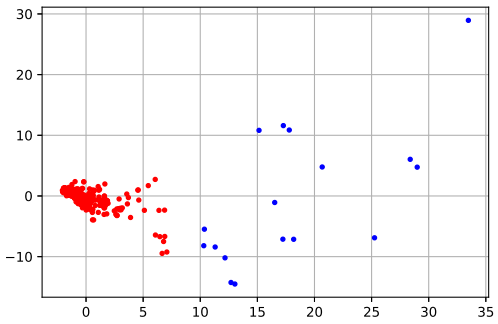

Fig. 5. Clusters after applying k-means algorithm

To find number of clusters we implement elbow method but figure does not show concrete number of clusters. So we have applied k-means with number of clusters k equals 2. Using the ”elbow method”, after the second point, the slope of the straight line connecting all points practically does not change. Thus it is not entirely clear justified the number of clusters equal to two.

2) DBSCAN: Using the DBSCAN algorithms, we did clustering and found that basically all points belong to two general classes and only 6% are outliers. We emphasize that this is a projection of data onto two main vectors and in n-dimensional space the picture looks different on Figure 6.

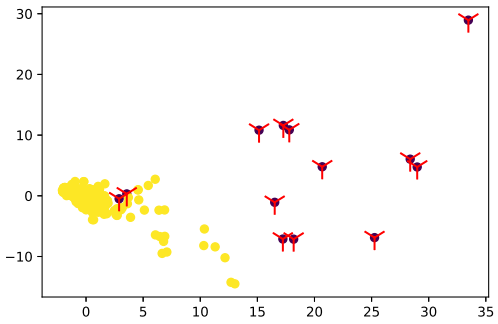

Fig. 6. Clusters with DBSCAN

C. Outliers and main cluster

1) 'Outliers': Observation which were determined as outliers by DBSCAN are the most interesting for us. We detected 58 outliers. It is not correct to call this observations outliers, because all metrics are collected correctly and there is no incorrect measurements. All this observations are big open-source projects. We can see that this list includes projects from Microsoft and Azure.

We have calculated statistics again, and for this 58 samples standard deviation is now comparable with mean. We applied k-means (number of clusters was determined by elbow method [28]) and DBSCAN clustering on this observations. These algorithms have shown the same results. Visualization is presented on Figure 7.

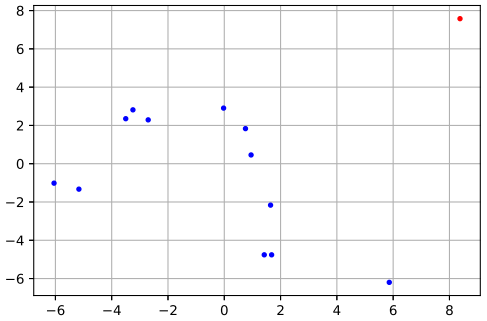

Fig. 7. Clustering of big projects

2) Main cluster: We have calculated statistics again for the rest of projects and standard deviation is now smaller or comparable with mean. We applied k-means clustering. Number of clusters is determined by elbow method and equals to 3. Results are presented on Figure 8.

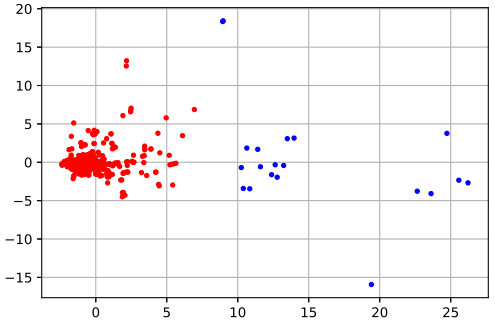

Fig. 8. Clustering on the rest of projects

VI. Conclusion

With the increasing value of knowledge presented by GitHub and one of the most important online source of software artifacts. We present a quantitative and qualitative analysis of GitHub repositories based on software project metrics. Firstly, we retrieved software metrics for 732 unique repositories. For each metric we calculated the main statistics: mean, median, mode standard deviation, 25%- and 75% quantiles. Overall metrics retrieved were 83 in total. Using the Kolmogorov-Smirnov test, we found out that the retrieved

metrics data is not normally distributed. Next we performed a correlation analysis. Based on the nature of the data we used to measure the correlation within the metrics. The insights from the correlations calculations was discussed in the context of software development context. After the test for normality and correlation measurements were performed, we clustered the data using two clustering methods : k-means and DBSCAN. Two clusters were obtained, one containing most of the repositories, namely individual projects of one owner, the second cluster contains almost all large projects included in the dataset. The number of clusters was determined using the elbow method. For the future we are planning to extend our study by increasing the number of projects to analyse and include code metrics to the list of metrics to analyse.

REFERENCES

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Tariq Ali, Sohail Asghar, and Naseer Ahmed Sajid. Critical analysis of dbscan variations. In *2010 international conference on information and emerging technologies*, pages 1–6. IEEE, 2010.

[3] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[4] Hudson Borges, Andre Hora, and Marco Tulio Valente. Predicting the popularity of github repositories. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 1–10, 2016.

[5] Hudson Borges, Andre Hora, and Marco Tulio Valente. Predicting the popularity of github repositories. PROMISE 2016, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Neil C Borle, Meysam Feghhi, Eleni Stroulia, Russell Greiner, and Abram Hindle. Analyzing the effects of test driven development in github. *Empirical Software Engineering*, 23(4):1931–1958, 2018.

[7] Fragkiskos Chatziasimidis and Ioannis Stamelos. Data collection and analysis of github repositories and users. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6. IEEE, 2015.

[8] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *Confidence intervals for the mean*, pages 341–360. Springer London, London, 2005.

[9] Michael Färber. *Analyzing the GitHub Repositories of Research Papers*, page 491–492. Association for Computing Machinery, New York, NY, USA, 2020.

[10] github. Github, 2020.

[11] S.C. Gupta and V.K. Kapoor. *Fundamentals of Mathematical Statistics*. Mathematical Sciences. Sultan Chand & Sons, 2020.

[12] Barbara Illowsky and Susan Dean. Introductory statistics. 2018.

[13] Abin Joy, Senthilkumar Thangavelu, and Amalendu Jyotishi. Performance of github open-source software project: An empirical analysis. 01 2018.

[14] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, page 92–101, New York, NY, USA, 2014. Association for Computing Machinery.

[15] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. An in-depth study of the promises and perils of mining github. *Empirical Software Engineering*, 21(5):2035–2071, 2016.

[16] Ben Lazarine, Sagar Samtani, Mark Patton, Hongyi Zhu, Steven Ullman, Benjamin Ampel, and Hsinchun Chen. Identifying vulnerable github repositories and users in scientific cyberinfrastructure: An unsupervised graph embedding approach. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6, 2020.

[17] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[18] Nora McDonald and Sean Goggins. Performance and participation in open source software on github. In *CHI'13 extended abstracts on human factors in computing systems*, pages 139–144. 2013.

[19] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. Curating github for engineered software projects. *Empirical Software Engineering*, 22(6):3219–3253, 2017.

[20] Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences*, 12, 2004.

[21] Marco Ortu, Giuseppe Destefanis, Daniel Graziotin, Michele Marchesi, and Roberto Tonelli. How do you propose your code changes? empirical analysis of affect metrics of pull requests on github. *IEEE Access*, 8:110897–110907, 2020.

[22] Cobra Rahmani and Deepak Khazanchi. A study on defect density of open source software. In *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 679–683, 2010.

[23] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[24] Leiming Ren, Shimin Shan, Xiujuan Xu, and Yu Liu. Starin: An approach to predict the popularity of github repository. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 258–273. Springer, 2020.

[25] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[26] Michael A Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American statistical Association*, 69(347):730–737, 1974.

[27] Nate Swanner. Big tech controls many major open source projects. is that a problem?, Aug 2019.

[28] M A Syakur, B K Khotimah, E M S Rochman, and B D Satoto. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. *IOP Conference Series: Materials Science and Engineering*, 336:012017, apr 2018.

[29] Dinesh Verma and Shishir Kumar. Prediction of defect density for open source software using repository metrics. *Journal of Web Engineering*, pages 293–310, 2017.

[30] Liu Wang, Ruiqing Li, Jiaxin Zhu, Guangdong Bai, and Haoyu Wang. A large-scale empirical study of covid-19 themed github repositories. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 914–923, 2021.

[31] Simon Weber and Jiebo Luo. What makes an open source code popular on git hub? In *2014 IEEE International Conference on Data Mining Workshop*, pages 851–855, 2014.

[32] Yunxiang Xiong, Zhangyuan Meng, Beijun Shen, and Wei Yin. Mining developer behavior across github and stackoverflow. In *SEKE*, pages 578–583, 2017.

[33] Yue Yu, Gang Yin, Huaimin Wang, and Tao Wang. Exploring the patterns of social behavior in github. In *Proceedings of the 1st international workshop on crowd-based software development methods and technologies*, pages 31–36, 2014.

[34] Yu Zhang, Frank F. Xu, Sha Li, Yu Meng, Xuan Wang, Qi Li, and Jiawei Han. Higitclass: Keyword-driven hierarchical classification of github repositories. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 876–885, 2019.

[35] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. The impact of continuous integration on other software development practices: a large-scale empirical study. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 60–71. IEEE, 2017.