Analysing Github Repositories Innopolis.AS.2022.Team5

Alaa Aldin Hajjar
Innopolis University
Innopolis, Russia
a.hajjar@innopolis.university

Odilbek Umaraliev
Innopolis University
Innopolis, Russia
o.umaraliev@innopolis.university

1 Introduction

In this project we aim to preform some statistical methods to do analysis on extracted metrics from 1000 github repositories and find any kind of relation between the metrics. Because of the metrics are from github that means the main things that are extracted are about the commits, forks, pulls, stars and etc. Using the right methods to analysis the data, and proper machine learning algorithem will give us a better understanding about the github repositories.

2 Methodology

we will divide the methodology to different parts:

2.1 Data Retrieval and Extraction:

In this step we will use a tool that retrieve an extract the data from 1000 github repositories using Docker container with Postgres database, we will do analysis will we retrieve the data to see the difference while the data is growing.

2.2 Data Preprocessing:

The data preprocessing step is divided to four main parts:
2.2.1 Removing null valued metrics because they have no effect.
2.2.3 Removing duplicated repos and mostly null values repos.
2.2.3 Replace null values with the mean value of the metric.
2.2.4 Scale the data using standard scaler (for clustering).

2.3 t test with Bootstrap:

After the data is preprocessed, we can apply t test so we wanted to understand the approximation to the sampling distribution of

$$t = (\bar{y} - \mu)/(s/\sqrt{n})$$

so we took a random sample y1, y2, ...,yn of size n from the population and computed the sample mean * then we selected random sample of size n with replace- ment from the sample and calculated the mean and std then we calculated

$$t = (\bar{y*} - \mu)/(s* /\sqrt{n})$$

then we repeated * 100 times then we sorted the t values and took value 25 and 75 and repeating the previous steps for different values of n

2.4 Kolmogrove-Smirnov Test For Normality:

Now we want to see if each metric has a normal distribution, so we will apply KS test which is a non-parametric statistical test that does not assume the shape of the distribution of the tested data. The main reason for this test is to know the normal distributed metrics (the ability to apply parametric tests) and not normal distributed metrics (there is no ability to apply parametric tests so we apply non-parametric tests) we will chose alpha value of 0.05.

2.5 Pearson Correlation:

To understand the data we need to check the correlation between the metrics and in this step we will use pearson correlation for normally distributed metrics which measures the linear correlation between two samples using the covariance and the variance.

2.6 Spearman Correlation:

If we have non normally distributed metrics we can use a non-parametric Spearman correlation to measure the rank of the correlation between two samples.

2.7 Difference between Spearman and Pearson Correlation:

In this part we will study the Difference between Spearman and Pearson Correlation.

2.8 Pearson Correlation with Bootstrap:

we will apply the Pearson correlation test with bootstrapping on normal distributed metrics that have more than 90% of correlation in Pearson, we will take a sample of the metrics with incremental size and for each sample we will make a n number of samples with replacement and do correlation test on them then sort the values and take 40% value and 60% value.

2.9 K-means Clustering:

In this step our aim is to split the data into clusters to give us more understanding of the metrics, so first we will apply PCA to do dimensionality reduction then k-means algorithm will work on partitioning a given data into k clusters, it starts by initialise random centroides for k clusters and calculate the distance between each data point and the centroides and based on this the centroides will change.

3 Results:

Note: For the full list of values we can visit the python notebook on github.
After cleaning and preprocessing the data we got 535 samples then we started with our methodology and we got the following results:

3.1 Results of t test with Bootstrap:

Here is a sample of the results for bootstraping with random choice with replacing and the size is 450:
commits count:

lower bound	upper bound	num of samples
-3.81	-2.18	40
-2.48	-1.25	80
-2.45	-1.36	120
-2.3	-1.26	160
-2.35	-1.44	200
-2.14	-1.35	240
-2.05	-1.23	280
-1.88	-1.14	320
-2.19	-1.35	360
-2.09	-1.0	400
-1.89	-1.24	440
-2.19	-1.15	480
-1.96	-1.16	535

wf avg failure duration

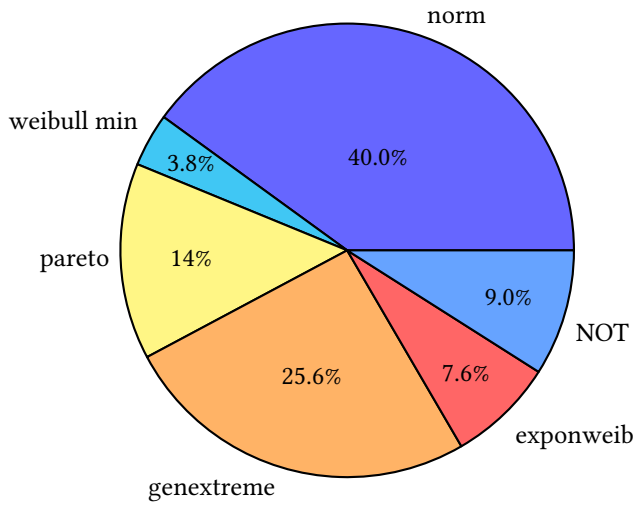
lower bound	upper bound	num of samples
-inf	-5.55	40
-3.95	-1.62	80
-11.58	-6.92	120
-8.5	-2.91	160
-98.84	-6.03	200
-67.32	-6.05	240
-32.42	-3.92	280
-12.24	-1.2	320
-12.38	-1.17	360
-26.32	-5.95	400
-12.34	-1.26	440
-26.65	-6.15	480
-93.69	-1.02	535

wf avg successes per day real

lower bound	upper bound	num of samples
-inf	-1.01	40
-4.03	-1.18	80
-2.81	-1.58	120
-2.47	-1.43	160
-2.6	-1.35	200
-2.18	-1.17	240
-1.98	-1.14	280
-2.71	-1.45	320
-2.28	-1.37	360
-2.69	-1.37	400
-2.46	-1.41	440
-2.15	-1.19	480
-2.04	-1.16	535

3.2 Results of Kolmogrove-Smirnov Test For Normality:

By applying previous Kolmogorov–Smirnov test for each feature to check the best p-value assuming that the distribution of feature samples is one of the following distributions "norm", "exponweib", "weibull-max", "weibull-min", "pareto" and "genextreme". The result that we found was the following



This is an example table:

Norm	
Features	p-value
commits_days_since_first	3.5e-11
issues_open	5.9e-70
issues_avg_comments	1.0e-58

Exponweib	
Features	p-value
commits_days_since_first	0.001
commits_avg_added	0.019
commits_avg_removed	0.00

Weibull_min	
Features	p-value
commits_total_lines_added	0.003
repo_size	0.01
repo_redme_length	2.1e-09

Pareto	
Features	p-value
stars_count	5.30e-14
contributors_count	8.50e-22
contributors_top_avg_commits	8.50e-22

Genextreme	
Features	p-value
commits_count	0.008
forks_count	0.005
repo_milestones	1.5e-118

3.3 Results of Pearson Correlation:

In this section we want to see which features are correlated more that 90%. We have this results (each set between two brackets are normally distributed metrics correlated together):

set 1
pulls avg files changed
pulls avg lines removed
pulls avg lines added

set 2
repo watchers
stars count

set 3
wf avg duration
wf avg failure duration

3.4 Results of Spearman Correlation:

In this section we want to see which features are correlated more that 90%. We have this results (each set of metrics between two brackets are correlated together):

set 1
pulls avg lines added
pulls total lines added
pulls max created per day
pulls avg created per day real
pulls count
pulls avg title length
pulls avg files changed
pulls avg Commits
pulls total lines removed
pulls avg lines removed

set 2
commits total lines added
commits total lines removed

set 3
repo workflows
wf count
wf avg success duration
wf avg successes per day real
wf avg duration
wf avg failure duration

set 4
repo age days
commits days since first

set 5
repo network members
forks count

set 6
issues max per day
issues count

set 7
contributors top avg commits
contributors top avg participation week

set 8
stars count
repo watchers

set 9
releases avg title length
releases count

set 10
releases avg assets size
releases avg assets
releases avg assets downloads
releases total downloads

3.5 Results of difference between Spearman and Pearson Correlation:

Max difference for set1 is: 0.123 Difference for set2 is: 0.002 Difference for set3 is: 0.029

3.6 Results of Pearson Correlation with Bootstrap:

Here is the results for bootstrapping with random choice with replacing and the size is 525:
issues labels, issues total comments, corr=0.92 :

lower bound	upper bound	num of samples
0.92	0.94	105
0.86	0.90	210
0.90	0.92	315
0.92	0.93	420
0.92	0.93	525

pulls avg lines added, pulls avg lines removed, corr=0.95

lower bound	upper bound	num of samples
0.52	0.63	105
0.92	0.96	210
0.14	0.17	315
0.89	0.96	420
0.86	0.95	525

wf avg duration, wf avg failure duration, corr=0.99

lower bound	upper bound	num of samples
0.97	0.63	105
0.97	0.96	210
0.96	0.17	315
0.99	1.00	420
0.99	1.00	525

4 Discussion:

We will discuss each part of the results and try to understand it.

4.1 Discussion of t test with Bootstrap:

From the examples that we have we can see that in commits count the value of t didn't converge well so we need to collect more data to represent the population and for wf avg successes per day real we can see that it is not improving and also for wf avg failure duration we can see weird behavior of the t value which is not understandable.
We chose this number of tables because they can represent the main idea

4.2 Discussion of Kolmogrove-Smirnov Test For Normality:

we can see that 40% of the metrics is distributed normally and 9% does not belong to any distribution with alpha value of 0.05 so for the normally distributed metrics we applied Pearson Correlation and for the non distributed metrics we applied Spearman Correlation.

4.3 Discussion of Pearson Correlation:

By looking to the correlation sets that we got in the results, in set 1 we can see a logical correlation between the issues total comments and issues labels, and if we looked to the second set we can see also a correlation between the avg add, removed lines and avg files changed because and add or remove to the lines will do changes on the files, and we can also see in set 3 that the more repo watchers that a repo have the more likely the number of stars will increase.

4.4 Discussion of Spearman Correlation:

In this part the main things that we can see from set 8 that the more network members that the repo has the more forks it has and this make sense.
We also can see that each family of metrics are correlated with each others like (commits, workflows, issues contributors pulls, etc)

4.5 Discussion of difference between Spearman and Pearson Correlation:

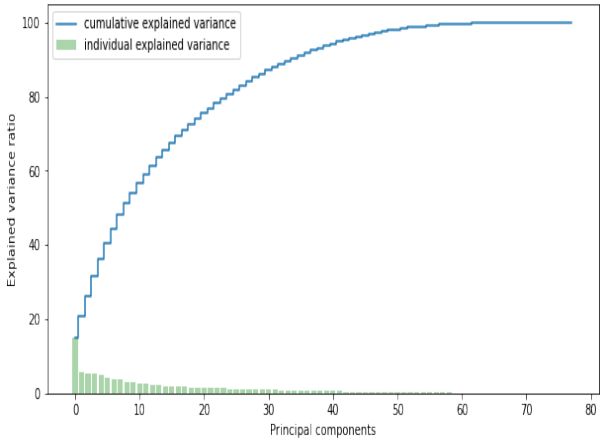
We clearly can see that the difference between Spearman and Pearson results is a small, which support our normality test and give us more confidence that the correlation results are true.

4.6 Discussion of Pearson Correlation with Bootstrap:

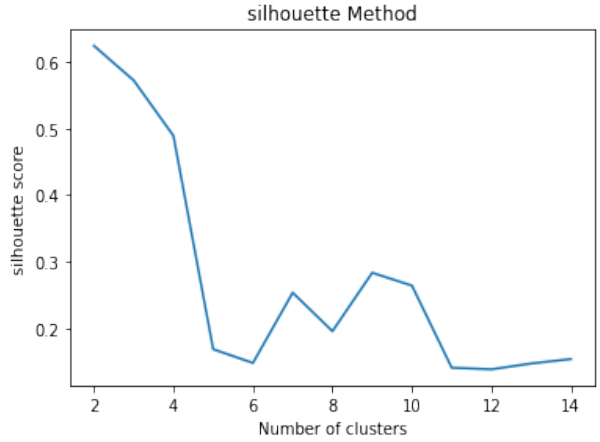
we can clearly see that the value of the correlation is between the lower bound and the upper bound which means our data is good enough for the correlation test, and this support our Pearson correlation test results.

5 K_means Clustering

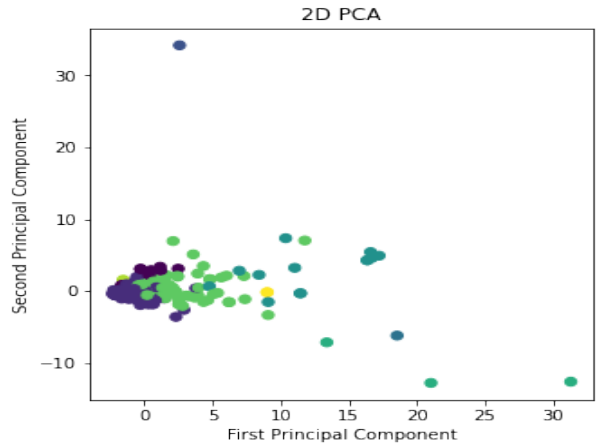
In this part we will try to cluster the repos, so the first step is to scale the data and apply PCA on it:



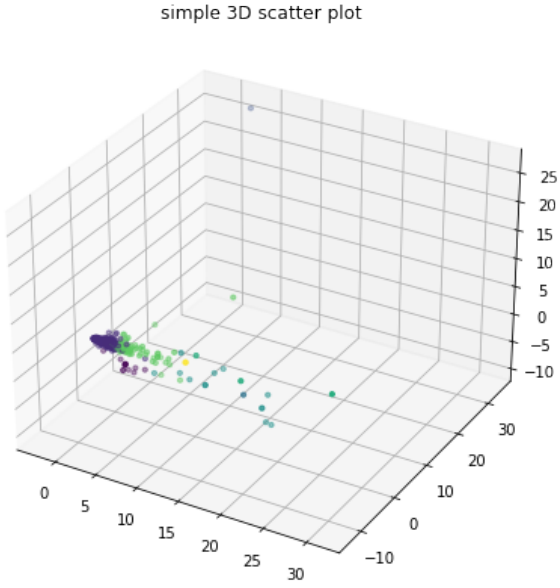
We can see with some linear combinations between columns, we can get more than 90% of the info with 40 features and with 60 features we can get more than 98% of info which means that there is a lot of Correlation in our data and there is some features that hold near to 0% of data so we will reduce the dim to 60 and then we will choose the number of clusters using silhouette_score and by looking to the figure we can see that 9 is some how good number for clusters.



After applying K_means with 9 cluster we got this results:



and for 3D visualization:



and after checking the number of repos in each cluster we can see:

cluster number	number of ropos
0	12
1	401
2	1
3	1
4	13
5	3
6	101
7	1
8	2

after this result we can see that the repos are one of two kinds based on the clustering results and if we look to the clusters centroides and go to github to check them we can see for cluster number 1 that the repos in this cluster are not popular and the stars and watch count are low and for cluster number 6 we can see the repos there are more popular with higher watch count and more stars and forks.

6 Conclusion

after we see the results of Spearman and Pearson correlation test that was supported by the bootstrapping and we also measured the difference between Spearman and Pearson correlation in normally distributed metrics which was very low (we did Pearson correlation test for only normally distributed metrics after applying Kolmogrove-Smirnov test for normality), we can conclude that repos with high number of watchers will have a high stars count, also the correlation between the pulls metrics tells us that the more lines are added the more changes we are going too see the more removed lines we will see which will increase the average number of commits need to do it, and this will increase in the

big repos, add to this that the number of network members will have high impact on the number of fork counts, at the end we have a clear view on the github repos and what each metric can change in this repo. an example that support our conclusion: https://github.com/andela/bp-esa-frontend

References

[1] Frank Konietschke and Markus Pauly. Bootstrapping and permuting paired t-test type statistics. *Statistics and Computing*, 24(3):283–296, 2014.

[2] Joost CF De Winter, Samuel D Gosling, and Jeff Potter. Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological methods*, 21(3):273, 2016.

[3] Manfred Mudelsee. Estimating pearson's correlation coefficient with bootstrap confidence interval from serially dependent time series. *Mathematical Geology*, 35(6):651–665, 2003.

[4] Shi Zhong, Taghi M Khoshgoftaar, and Naeem Seliya. Analyzing software measurement data with clustering techniques. *IEEE Intelligent Systems*, 19(2):20–27, 2004.