Software engineer profile recognition through application of data mining techniques on GitHub

Inglezos Charalampos
Department of Electrical and Computer Engineering
Aristotle University of Thessaloniki
Thessaloniki, Greece
inglezos@ece.auth.gr

Abstract—**This assignment is about analyzing and processing a variety of open-source software repositories data that have been already mined and trying to categorize them into groups, depending on their characteristics and features. More specifically, the issue is to observe among the data the profiles that the software engineers (contributors) can have and further divide them into subcategories, extracting many more details about their skills and traits, using some metrics and thus, hopefully, develop approximately their personal repository profile.**

Keywords*—*Pattern Recognition; RStudio; Clustering; Engineer Role Recognition; mining software repositories; metrics; Github; Dev; Ops; DevOps

I. Introduction

In this paper, data have been collected from multiple Github repositories with the help of Github API, data from which I am trying to extract possible profiles of the software engineers that have contributed to, analyzing and implementing various clustering techniques in RStudio (R language). First and foremost, I hope to observe at least three categories or profiles of contributors: Users occupied with pure Software Development (Dev), pure Software Operations (Ops) and DevOps, a combination of the two above categories. It is also possible to find some outlier values or some other categories of different profile characteristics. Finally, for each category (Dev, Ops, DevOps), I examine the different features and attributes each group has and therefore I can further describe in more detail not only if a contributor belongs to any of the three categories and which one, but also if he has some other features (i.e. if he is productive, responsive, etc) and to what extent.

II. RESEARCH OVERVIEW

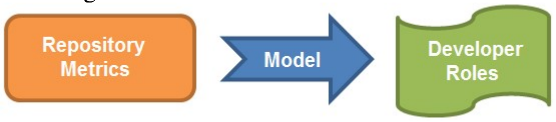
In order to perform the data analysis, first I have to examine the data that have been collected, their format, realize the metrics that probably I am going to use, see their summary statistics and how their values behave, so as to get a general picture of what I am dealing with. A first thought was to try to visualize them, but that was impossible since there are 25 dimensions (these are the metrics/attributes/columns).

Therefore, the first questions and problems I faced was to understand the nature of the given data and then to preprocess them in order to bring them to a better value format, since many values could be missing or some outliers could exist, for multiple metrics (attributes). The next problem was to choose the proper clustering algorithms so as to divide the data into the three categories mentioned above (Dev, Ops, DevOps), but also the proper algorithm to divide further each category into subcategories, extracting features of the formed groups.

III. SYSTEM DESIGN

A. System Overview

The system I am going to analyze has the following overall data flow diagram:



The repository metrics are the columns of the data file (in .csv format), which are the different characteristics that I am considering to analyze the data. The model is the object produced by the clustering algorithm and includes its parameters, the different tests and my examining choices. And ultimately, the developer roles are the three discrete roles I am hoping to observe in the data, the Dev the Ops and the DevOps.

B. Data Preprocessing

Examining the data, I easily found (i.e. command *summary()*) thousands of NA values; this means they are missing values and they have not been documented or observed and this is a major problem that I had to deal with. For that purpose I implemented data preprocessing analysis in three stages, for the initial dataset of 8010×27 dimensions, named *dataset*.

First of all, in stage one, I delete all the records/lines that have zero activity period, since these cases mean that the users/contributors were completely inactive and they can't represent the data at all. Now, the initial dataset becomes 5704×27 (dimensions) $\rightarrow$ *dataset_2*.

In the second stage, for the ”inactive period within active period” metric, I find the numerical value that corresponds to the 90% quantile (command *quantile()*) and I reject all the records with higher value for this metric, since such highly inactive contributors consist extreme outliers for the dataset and probably aren't representative of the data. Now the dataset has 5137×27 dimensions → *dataset_3*.

In stage three, initially, I create dataset_4 from the numerical columns of dataset_3 (I reject the first two ones, the contributor's login name and the repository name). Then, I see that two metrics, "tot_additions" and "tot_deletions" have a very extreme max value compared to the mean. So, I decide to apply again some quantile restrictions for these columns, for 99.9% to both of them and keep the records that are less or equal to the value corresponding to the 99.9% quantile → *dataset_5*, with dimensions 5130×25.

After that, I try to clear the data from all the still missing values (3 columns left with NA values). With the help of a utility function I created, the "impute_NA", I impute with 0 the NA values of the "average_issues_comments_length" and "average_comments_per_issue" columns (assuming the worst case scenario) with resulting dataset the *dataset_6* and the remaining NAs, of the "average_time_to_close_issues" metric, with the MICE package (commands *mice(), complete()*) → *dataset_7* (5130×25).

C. Model Construction

After having preprocessed the data now I can apply a variety of clustering algorithms to try to categorize them into Dev, Ops and DevOps. Many different algorithms were tried and many versions of the datasets were tested. Initially the base dataset was dataset_7, but finally, as I will later describe thoroughly, the dataset_final will be the one I ultimately chose. Also, I tried scaling the datasets, but the results (boxplots visualization of the metrics and the silhouette coefficient) were worse than the original. I implemented DBSCAN, kmeans, hclust (hierarchical), diana (divisive hierarchical), fanny (fuzzy hierarchical), agnes (agglomerative hierarchical) for *euclidean*, *manhattan* and *minkowski* metrics and *complete* and *average* methods. For my convenience for the implementation, but also for model statistical and comparative reasons, I used the *factoextra*, *fpc*, *clvalid* and *eclust* packages.
Also, I have to note that I tried PCA and ISOMAP just to see whether I could proceed with their results, but I didn't get anything useful from these methods, which are mostly for visualization and dimension reduction, since we have too many dimensions and additionally ISOMAP is very resource-demanding on this data scale.

IV. EVALUATION

A. Evaluation Methodology

Having created many models, I evaluated them thoroughly. DBSCAN was the most unsuccessful algorithm (in terms of proper dividing the data into groups and silhouette, for a variety of values for the *eps* and *minPts* parameters) and this is more or less justified, since the data have many different densities. After many days of research and testing the above algorithms for various parameters, I found that hclust and agnes behave similarly so I kept hclust (due to its slightly higher silhouette) for the comparison tests. Working still on dataset_7, I tried all the rest algorithms, with the criteria for which is better than the others being first the boxplots visualization (if the results for each metric of the dataset_7 were properly clustered and especially if at least three groups/categories were discerned) and secondly the silhouette factor, as well as the cohesion and separation. But mostly, the important factor were the boxplots, considering silhouette higher than 0.6 value limit.
At this point, I didn't know how many clusters I should try to create, so I tried to implement gap statistics for k, using the aforementioned eclust package. But the problem I faced was that it needed lots of bootstrapping iterations and in my computer it took many hours just for one value of k and for specific parameters. I think it is obvious that for a parametric analysis both for k and for other parameters and for each algorithm it would take days or worse weeks to finish and for me to have a complete result. But even in that way, the result would rely only on some metrics (silhouette mostly) to indicate which algorithm is the best and for what k, which would probably be wrong, since as I said the criteria were the boxplots visualization and secondly the good clustering formation. Therefore, I tried some values for k manually for specific parameters, trying to get some preliminary results.

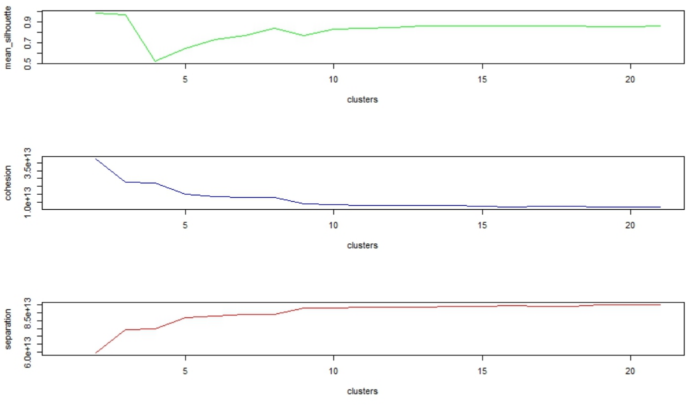


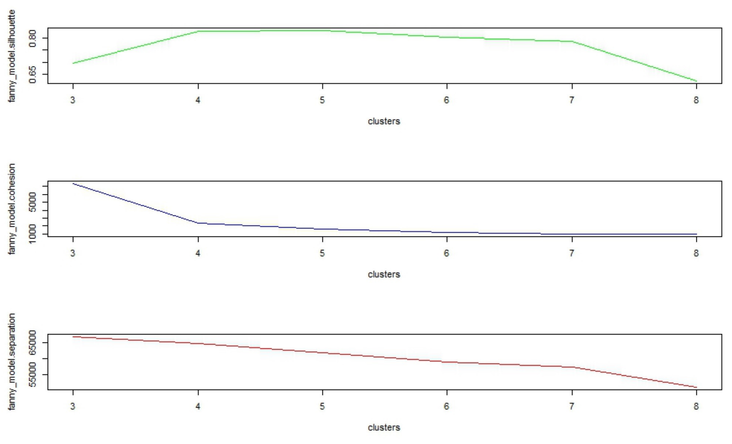
Fig. 1. Kmeans Performance

<u>Note:</u> *All the following test models can be further analyzed by the reader using the accompanying R data file, which includes all my models.*

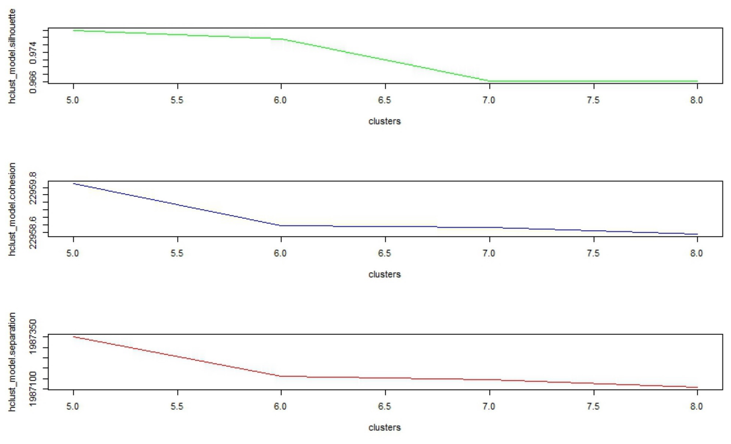
1) Kmeans: The kmeans algorithm shows good silhouette performances, not that good as the other algorithms, but exhibits better boxplots behavior, especially for k = 7.

2) Fuzzy Hierarchical (fanny): The fuzzy hierarchical algorithm (fanny) has good silhouette for k = 5, 6, 7, but the boxplots analysis was somehow poor and showed that most of the features were accumulated in three specific groups, while the rest have only a few.

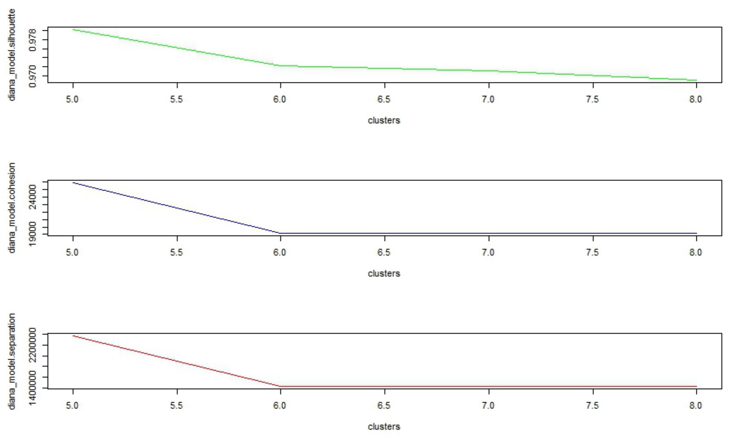
3) Divisive Hierarchical (diana): The divisive hierarchical (diana) has extremely high silhouette for the same k values,



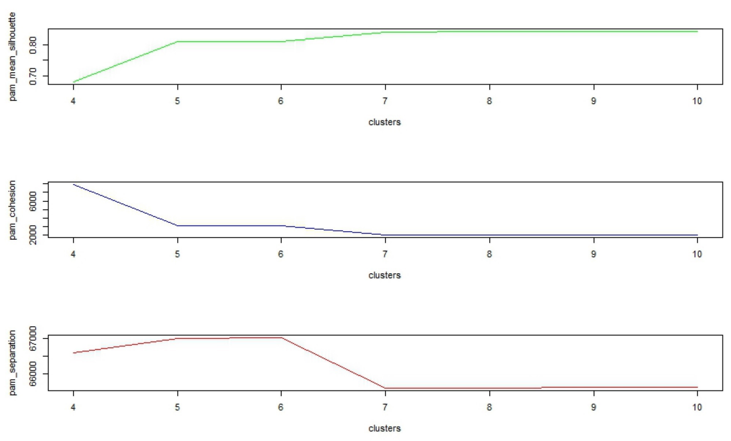
Fuzzy Hierarchical (fanny) Performance



Hierarchical (hclust/agnes) Performance



Divisive Hierarchical (diana) Performance



Partitioning Around Medoids (pam) Performance

Fig. 2. Performance results for the fanny, diana, hclust and pam algorithms.

but bad-formed clusters, except perhaps for k = 6.

4) Hierarchical (hclust/agnes): Finally, the hierarchical (hclust) shows same performance for the euclidean and minkowski metric, but the manhattan and specifically the average one, seems to be better and during the boxplot analysis, for k = 6.

Thus I opted for the hclust, manhattan, average, for k = 6 (nevertheless, the diana algorithm had similar results).

But soon, I realized that unfortunately all these were in vain and wrong, since the clusters that were formed with hclust (and also with diana) were disproportionate in the number of observations, and from 5130, the almost 5000 records had been accumulated into only one cluster, while the others clusters had only a few, at most some decades of records. That was obviously not desirable.

After much more research, I decided to change my working dataset. After having tried scaling, different metrics and after having implemented a p-value (null hypothesis) test that did not work well enough, I decided to keep only specific metrics/columns from the 25, since they were indicative enough of the information and thus I could get some useful overall results by using only these. I kept all the *issues* metrics, the *tot_additions* and *tot_deletions*, *commits_authored*, *total_lines_of_code_changed* and the *violations* metrics. I think these can be enough to categorize the data into the three groups. Thus, the working dataset now becomes the *dataset_final* $\rightarrow$ of 5130×12 dimensions. I implemented once again the hclust algorithm but again it is inefficient as described. So, after research for other algorithms, I chose to apply the pam (partitioning around medoids) algorithm, for euclidean metric and the resulting model is the pam_model.

B. *Final Model: Partitioning Around Medoids (pam)*

The problem now is not that difficult to be handled, since only the half almost records are accumulated in one cluster, while the other clusters are efficiently formed. To overcome this problem, in the one bigger cluster, I implemented once again the pam algorithm and after many tests I decided to use 8 groups (k = 8) for the first pam and another 8 subgroups for the second pam applied to the bigger cluster. This choice was made to further subdivide the bigger cluster efficiently, as the first pam for fewer clusters wasn't very effective towards this purpose. Finally, the initially bigger cluster with over 5000 observations became of around 3000 (*first pam*) and now of 1700 approximately (*second pam*).

Altogether, 15 clusters/groups are ultimately formed. Using the pam model, I will present in the next subsection, for each cluster and for each column/metric, the results of the boxplots, characterizing the clusters as *Extra Low (EL)*, *Low (L)*, *Medium (M)* and *High (H)*. Then finally, I will assign every cluster into one category, either *Dev (D)* or *Ops (O)* or *DevOps (DO)* or *Uncategorized/Unknown (U)*.

C. *Evaluation Results*

Using the pam algorithm and the clustering data it provided, we can see some indicative boxplots visualization plots for some metrics, for the dataset_final, in *Fig. 3* and *Fig. 4*.

We observe that few clusters aren't formed in an optimal way, but as we will see in the summary table, this doesn't produce errors in discerning the desirable three contributors categories. Also, some clusters that in the pictures don't seem to be very well formed, if we zoom into the boxplot and change the y-axis limits, we will see that they are indeed better formed, they just have lower values compared with the higher-valued clusters for the examined metric. The reader can realize this using the R data file accompanying this assignment report. All these realizations are summarized in the table(*TABLE I*), where I have quantified the boxplots results and have interpreted them into essentially three categories; Low, Medium and High as previously described.

As we can observe from this table, we have found what we were looking for! The desired three categories (Dev, Ops, DevOps) can be discerned in the data. The attributes in the last column ("Category") were assigned in my personal opinion, examining the values (*EL*, *L*, *M*, *H*) for each cluster and for each metric. Maybe some readers will think differently to assign the categories to each cluster.

Another important thing we can observe is that some clusters are recognized by two categories (i.e. DevOps/Dev, U/DevOps, U/Ops). The dual characterization means that the first one is more evident, while partially the second one is valid. This applies also to the metric values in the table (i.e.: L/M means mostly Low and partially Medium). I believed it was right to assign the DevOps/Dev cluster to DevOps and to combine into one U cluster the three U/DevOps and U/Ops. So, the final pam model is the pam_final, which now has 13 distinct clusters, summarized in *TABLE II*.

Clusters \ Metrics	Comments length	Issues closed	Issues opened	Issues participated	Issues closed per day	Comments per issue	Total additions	Total deletions	Commits	Total lines of code changed	Violations added	Violations eliminated	Category
1	L	L	L	L	L	L/M	L/M	L	L	L/M	L/M	L/M	U/DevOps
2	EL	L	L	L	L	L/M	L/M	L	L	EL	EL	EL	U/DevOps
3	L	EL	EL	EL	EL	M	L	L	EL	EL	EL	EL	Ops
4	L	EL	EL	EL	EL	M	EL	EL	EL	EL	L	L	Ops
5	L	L	L	L	L	L/M	L	L	L	L	L/M	L	U/Ops
6	EL	EL	EL	EL	EL	M	EL	EL	EL	EL	EL	EL	Ops
7	M	EL	EL	EL	EL	M	EL	EL	EL	EL	EL	EL	Ops
8	M	L	L	M	M	L	L	L	L	M	M	M	DevOps/Dev
9	L	M	M	M	M	L/M	M	M	M	H	H	M	DevOps
10	L	M	M	M	M	L	M	L	L	H	H	M	Dev
11	H	EL	EL	EL	EL	H	L	L	EL	L	L	L	Ops
12	L	H	H	H	H	L	H	H	H	M	M	H	DevOps
13	L	M	M	M	M	L	M	M	M	H	H	H	Dev
14	H	M	H	M	M	H	H	M	L	M	M	L/M	DevOps
15	M	M	M	M	M	L	H	H	M	M	M	M	Dev
EL = Extra Low, L = Low, M = Medium, H = High, U = Uncategorized/Unknown													

TABLE I. ASSIGNING EACH CLUSTER TO ONE OF THE FOUR CATEGORIES: DEV, OPS, DEVOPS AND UNCATEGORIZED/UNKNOWN.

Clusters	1	2	3	4	5	6	7	8	9	10	11	12	13
Category	U	Ops	Ops	Ops	Ops	DevOps	DevOps	Dev	Ops	DevOps	Dev	DevOps	Dev
Number of Contributors	210	167	342	1622	659	9	65	205	1788	9	43	6	5

TABLE II. SUMMARY OF THE FINAL CLUSTERS CREATED.

D. *Further Profile Analysis*

Now that we have found the three contributors profiles, we can try to further analyze each one of them in detail, extracting specific features about these Github users, with the help of the metrics given. To do so, I created three subdatasets, one for the Dev, one for the Ops and one for the DevOps. I implemented all the described above algorithms to these three datasets and tried to subdivide them into groups, with ultimate goal to extract useful information about their personal profile as engineers. I found that once again the

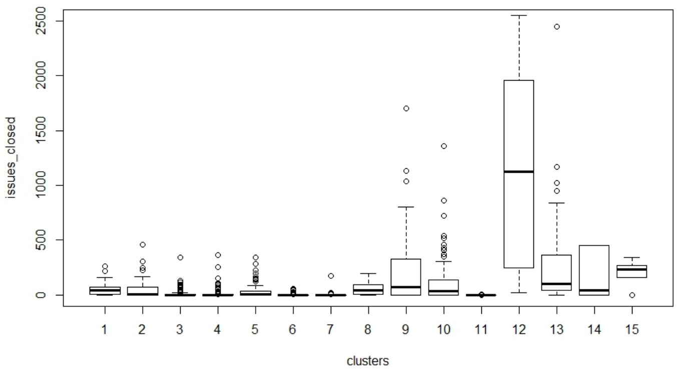


Fig. 3. Boxplot for the Issues Closed, using pam

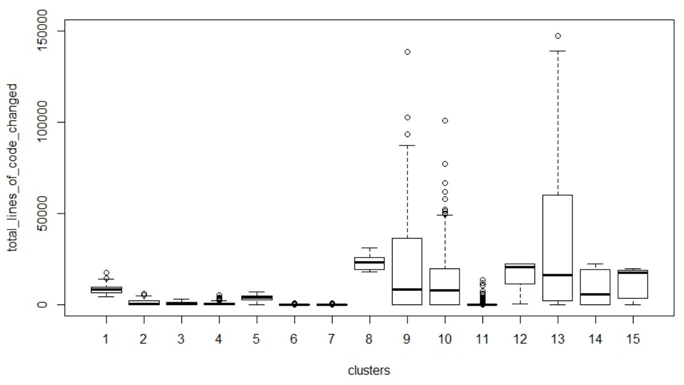
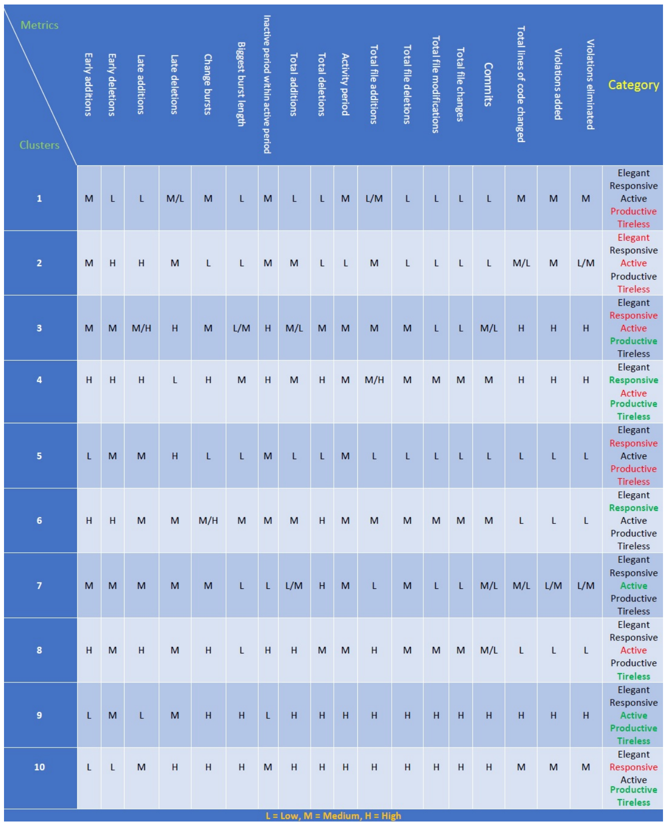


Fig. 4. Boxplot for the Total Lines of Code Changed, using pam



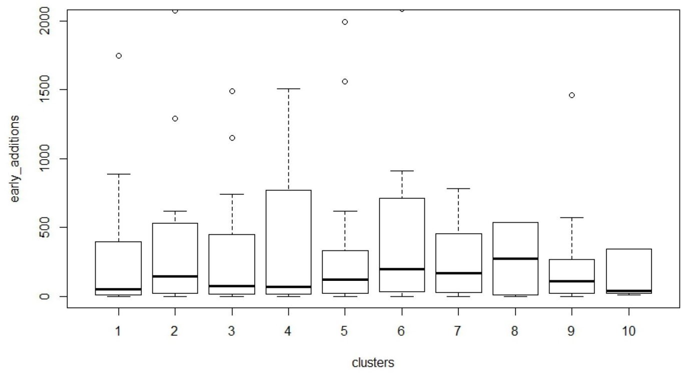
Clusters \ Metrics	Early additions	Early deletions	Late additions	Late deletions	Change bursts	Biggest burst length	Inactive period within active period	Total additions	Total deletions	Activity period	Total file additions	Total file deletions	Total file modifications	Total file changes	Commits	Total lines of code changed	Violations added	Violations eliminated	Category
1	M	L	L	M/L	M	L	M	L	L	M	L/M	L	L	L	L	M	M	M	Elegant Responsive Active Productive Tireless
2	M	H	H	M	L	L	M	M	L	L	M	L	L	L	L	M/L	M	L/M	Elegant Responsive Active Productive Tireless
3	M	M	M/H	H	M	L/M	H	M/L	M	M	M	M	L	L	M/L	H	H	H	Elegant Responsive Active Productive Tireless
4	H	H	H	L	H	M	H	M	H	M	M/H	M	M	M	M	H	H	H	Elegant Responsive Active Productive Tireless
5	L	M	M	H	L	L	M	L	L	M	L	L	L	L	L	L	L	L	Elegant Responsive Active Productive Tireless
6	H	H	M	M	M/H	M	M	M	H	M	M	M	M	M	M	L	L	L	Elegant Responsive Active Productive Tireless
7	M	M	M	M	M	L	L	L/M	H	M	L	M	L	L	M/L	M/L	L/M	L/M	Elegant Responsive Active Productive Tireless
8	H	M	H	M	H	L	H	H	M	M	H	M	M	M	M/L	L	L	L	Elegant Responsive Active Productive Tireless
9	L	M	L	M	H	H	L	H	H	H	H	H	H	H	H	H	H	H	Elegant Responsive Active Productive Tireless
10	L	L	M	H	H	H	M	H	H	H	H	H	H	H	H	M	M	M	Elegant Responsive Active Productive Tireless

L = Low, M = Medium, H = High

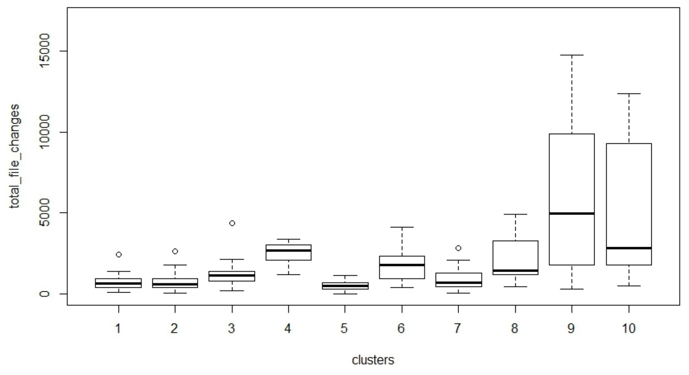
TABLE III. Dev Profile Boxplot Analysis.

pam method gives the best results, according to the boxplot analysis (the silhouette coefficient is secondary, because I don't expect profound cluster isolation, since these clusters that will be further created are part of a cluster already created from clustering). The results for each category (Dev, Ops, DevOps) is presented thoroughly in *TABLE III - TABLE V*.

1) Dev Profile Analysis: I created the dev_detailed_dataset from the dataset_7 for all the records that have pam clustering



Dev Boxplot for Early Additions.



Dev Boxplot for Total File Changes.

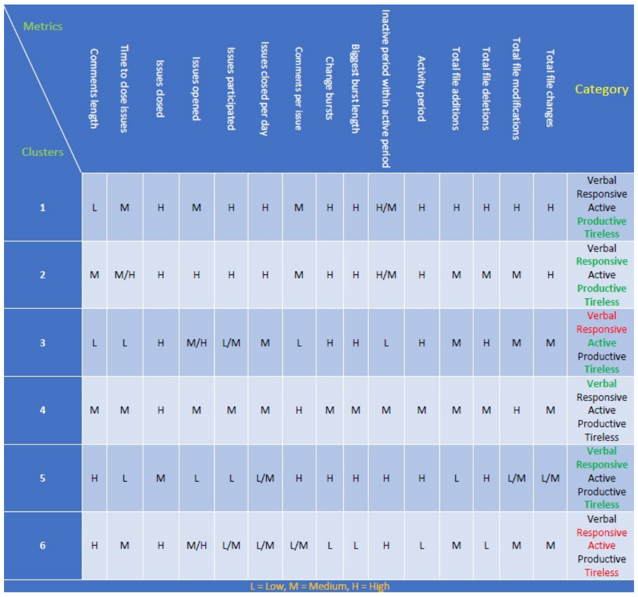
Fig. 5. Dev Boxplots for specific metrics.

= 8, 11, 13. Then I created the dev_detailed_dataset_filtered from dev_detailed_dataset, rejecting Ops metrics (all the issues metrics) $\rightarrow 253\times18$. With pam, I found that for the later boxplot analysis, it is best to choose first k = 3 and the bigger cluster to subdivide it again with pam for k = 8; totally 10 dev clusters. The boxplot analysis for the Dev profile is shown in detail in *TABLE III*. With **black** text color I symbolize the medium level of the feature, with red the little and with green the high; for example Elegant means that this subgroup of Dev is little elegant (in their coding performance). In *Fig. 5* we can see some indicative boxplots for some metrics of the Dev submodel. The attributes assigned to column "Category" are the following:

a) Elegant: This depends on the violations and it is high if the violations added are low and the violations eliminated are high.

b) Responsive: This depends on the (low) late and the (high) early additions/deletions respectively.

c) Active: This depends on the active period and mostly on the (low) inactive period.



Clusters \ Metrics	Comments length	Time to close issues	Issues closed	Issues opened	Issues participated	Issues closed per day	Comments per issue	Change bursts	Biggest burst length	Inactive period within active period	Activity period	Total file additions	Total file deletions	Total file modifications	Total file changes	Category
1	L	M	H	M	H	H	M	H	H	H/M	H	H	H	H	H	Verbal Responsive Active Productive Tireless
2	M	M/H	H	H	H	H	M	H	H	H/M	H	M	M	M	H	Verbal Responsive Active Productive Tireless
3	L	L	H	M/H	L/M	M	L	H	H	L	H	M	H	M	M	Verbal Responsive Active Productive Tireless
4	M	M	H	M	M	M	H	M	M	M	M	M	M	H	M	Verbal Responsive Active Productive Tireless
5	H	L	M	L	L	L/M	H	H	H	H	H	L	H	L/M	L/M	Verbal Responsive Active Productive Tireless
6	H	M	H	M/H	L/M	L/M	L/M	L	L	H	L	M	L	M	M	Verbal Responsive Active Productive Tireless

L = Low, M = Medium, H = High

TABLE IV. OPS PROFILE BOXPLOT ANALYSIS.

d) Productive: This is an overall factor that is associated with many metrics ((high) total file additions/deletions/modifications/changes, (high) total lines of code changed and (high) early and (low) late additions/deletions.

e) Tireless: This depends on the (high) change bursts and the biggest burst length metrics.

2) Ops Profile Analysis: I created the ops_detailed_dataset from the dataset_7 for all the records that have pam clustering = 2, 3, 4, 5, 9. Then I created the ops_detailed_dataset_filtered from dev_detailed_dataset, rejecting Dev metrics (total lines of code changed, commits, total additions/additions, early/late additions/deletions and the violations metrics) $\rightarrow$ 4578$\times$15. With pam, I found that for the boxplot analysis, it is best to choose k = 6 and not to divide further any cluster produced (the boxplot results, as well as the clusters formation in this case was not as good as the rest two specific profile cases). The boxplot analysis for the Ops profile is shown in detail in *TABLE IV*. In *Fig. 6* we can see some indicative boxplots for some metrics of the Ops submodel. The attributes assigned to column "Category" are the following:

a) Verbal: This depends on the (high) comments per issue and on the comments length.

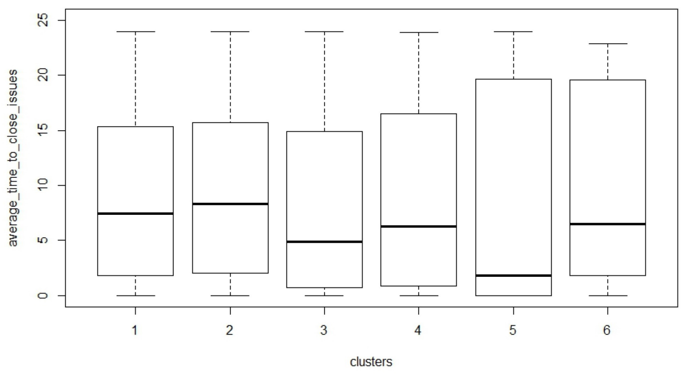
b) Responsive: This depends on the (low) time to close issues, the (high) issues closed per day and (high) comments per issue.

c) Active: (Same as described above)

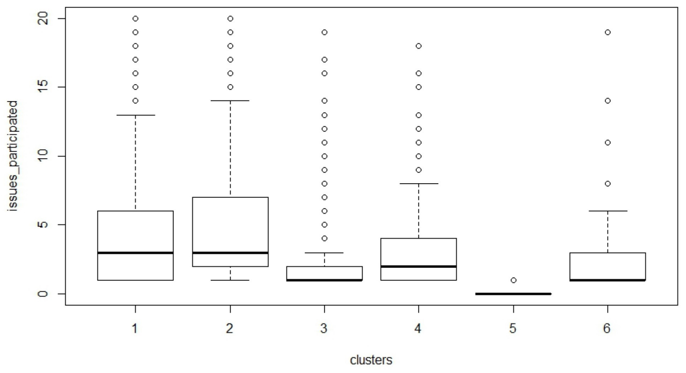
d) Productive: This is an overall factor that is associated with many metrics ((high) the file and the issue metrics).

e) Tireless: (Same as described above)

3) DevOps Profile Analysis: Similarly, I created the devops_detailed_dataset from the dataset_7 for all



Ops Boxplot for Average Time to Close Issues.



Ops Boxplot for Issues Participated.

Fig. 6. Ops Boxplots for specific metrics.

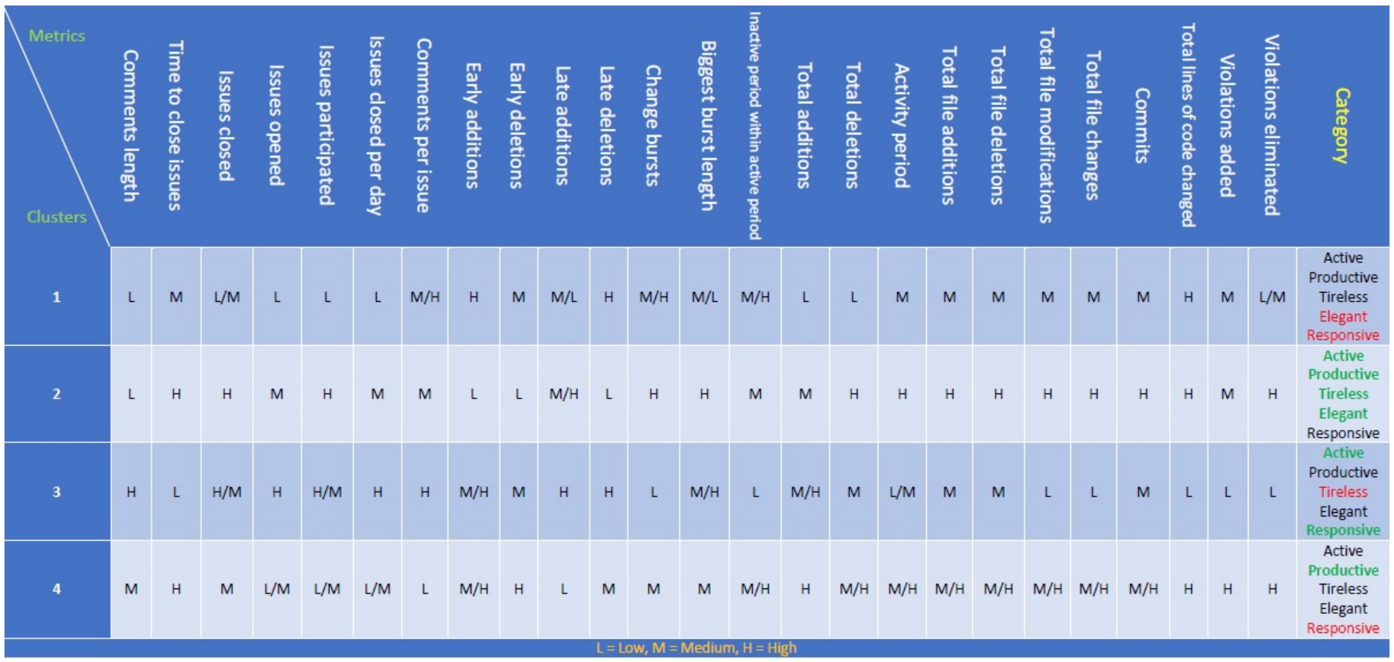
the records that have pam clustering = 6, 7, 10, 12. Then I created the devops_detailed_dataset_filtered from devops_detailed_dataset, keeping all the metrics $\rightarrow$ 89$\times$25. With pam, I found that for the boxplot analysis, it is best to choose k = 4 (for higher k value the silhouette coefficient drops rapidly) and not to divide further any cluster produced. The boxplot analysis for the DevOps profile is shown in detail in *TABLE V*. In *Fig. 7* and *Fig. 8* we can see some indicative boxplots for some metrics of the DevOps submodel. The attributes assigned to column "Category" have been already described above, combining the metrics they depend on, from both Dev and Ops metrics and for this profile are the following:

a) Active: (Same as described above)

b) Productive: This is an overall factor that is associated with the respective metrics of both Dev and Ops profile (i.e. (high) issues metrics, total lines of code and total additions, etc)

c) Tireless: (Same as described above)

d) Elegant: (Same as described above)



Clusters \ Metrics	Comments length	Time to close issues	Issues closed	Issues opened	Issues participated	Issues closed per day	Comments per issue	Early additions	Early deletions	Late additions	Late deletions	Change bursts	Biggest burst length	Inactive period within active period	Total additions	Total deletions	Activity period	Total file additions	Total file deletions	Total file modifications	Total file changes	Commits	Total lines of code changed	Violations added	Violations eliminated	Category
1	L	M	L/M	L	L	L	M/H	H	M	M/L	H	M/H	M/L	M/H	L	L	M	M	M	M	M	M	H	M	L/M	Active Productive Tireless Elegant Responsive
2	L	H	H	M	H	M	M	L	L	M/H	L	H	H	M	M	H	H	H	H	H	H	H	H	M	H	Active Productive Tireless Elegant Responsive
3	H	L	H/M	H	H/M	H	H	M/H	M	H	H	L	M/H	L	M/H	M	L/M	M	M	L	L	M	L	L	L	Active Productive Tireless Elegant Responsive
4	M	H	M	L/M	L/M	L/M	L	M/H	H	L	M	M	M	M/H	H	M/H	M/H	M/H	M/H	M/H	M/H	M/H	H	H	H	Active Productive Tireless Elegant Responsive

L = Low, M = Medium, H = High

TABLE V. DEVOPS PROFILE BOXPLOT ANALYSIS.

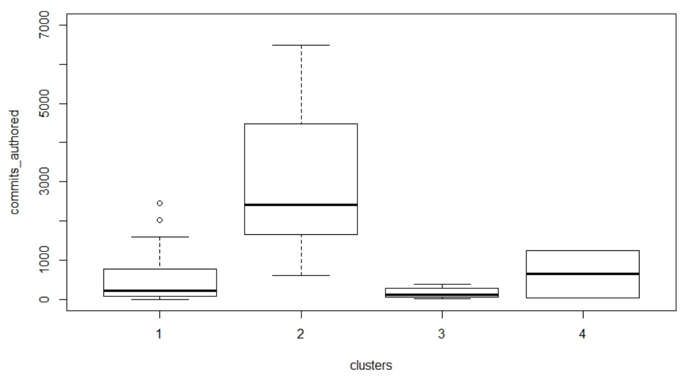


Fig. 7. DevOps Boxplot for Commits Authored.

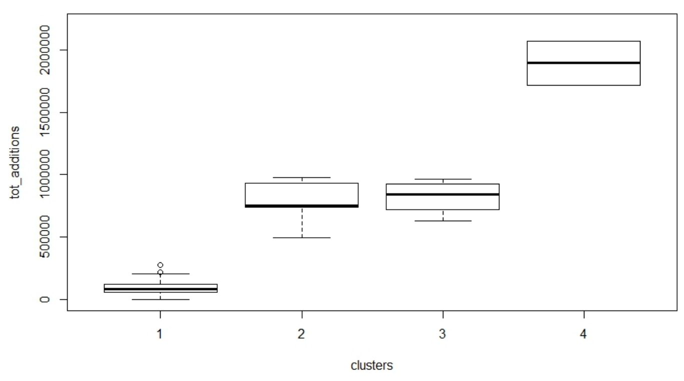


Fig. 8. DevOps Boxplot for Total Additions.

e) Responsive: This depends on the (low) time to close issues, the (high) issues closed per day, the (high) comments per issue and on the (low) late and the (high) early additions/deletions respectively.

V. CONCLUSIONS

In this assignment, from the detailed results we can certainly conclude that the three profiles were discerned indeed, while one other category was found that was mostly unknown but had traits of DevOps and partially of Ops.

We can also conclude that, first of all, the Ops are the vast majority of the records. Observing the three summary tables (*TABLE III - TABLE V*) for each profile, with regards to the features of each one, we realize that the Dev can have many attributes as their personal software engineer traits, varying from little productive and inactive to fully productive and tireless (which means they are very active, productive and responsive in a short period of time, as is indicated by the bursts metrics), while none of the Dev clusters are highly elegant (which is somehow expected, since it is very difficult someone's code writing to be characterized excellent or to eliminate constantly violations without often making new ones). The Ops have their respective features and they are never little productive, while the DevOps are always medium and highly active, productive too in general and sometimes highly elegant.

VI. FUTURE RESEARCH - IMPROVEMENTS

Obviously, there are probably some things that could be further improved. There are many other algorithms that could be tested and on other subdatasets than the original and for other parameter values. Also, the preprocessing could have been different and this could have led to different clusters and profile features. ISOMAP could be utilized at some extent, in order to visualize a priori some data metrics, which could maybe give us a better perspective of the data and their metrics relations.

This entails much better hardware and many more resources, perhaps using cluster-computers and some parallel coding in R, which could be used for running more algorithms and comparative tests to find more and more optimal solutions. In addition to the above, we could collect many more repositories in our initially given data and examine more metrics. Finally, the ops category/dataset can probably be even better clustered into ops features, depending on the algorithms and parameters that will be tested for.

ACKNOWLEDGMENT

The author would like to thank the postgraduate students of the ”Pattern Recognition” course for their help, support and guidance for this assignment.

REFERENCES

[1] Master Thesis of I. Zafeiriou (June 2017) : Software Engineer Profile Recognition using data mining analysis in Github repositories source codes and comments. https://cassiopia.ee.auth.gr/index.php/s/UWWGAE0q6xctn7g/download

[2] S. Bayati, D. Parsons, T. Susnjak and M. Heidary, ”Big data analytics on large-scale socio-technical software engineering archives,” 2015 3rd International Conference on Information and Communication Technology (ICoICT), Nusa Dua, 2015, pp. 65-69.

[3] F. Chatziasimidis and I. Stamelos, ”Data collection and analysis of GitHub repositories and users,” 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), Corfu, 2015, pp. 1-6.

[4] A. Giri, A. Ravikumar, S. Mote and R. Bharadwaj, ”Vritthi - a theoretical framework for IT recruitment based on machine learning techniques applied over Twitter, LinkedIn, SPOJ and GitHub profiles,” 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), Ernakulam, 2016, pp. 1-7.

[5] *https://dl.acm.org/citation.cfm?id=2597074*

[6] *https://blog.exploratory.io/analyzing-issue-data-with-github-rest-api-63945017dedc*

[7] *https://www.novoda.com/blog/github-data-mining-101/*

[8] *https://www.safaribooksonline.com/library/view/mining-the-social/9781449368180/ch07.html*