Application of software engineering role recognition techniques using knowledge mining techniques from open source repositories

Stergios Koukouftopoulos(AEM:8694),Zafeirios Mpampos(AEM:8732)
Electrical and Computer engineering
Aristotle University
Thessaloniki, Greece
stergios06@hotmail.com, z.mpampos@auth.gr

I. Introduction

Software development methodologies, or process models, attempt to describe the steps that should be followed along the way from conception to deployment of software. There are traditional approaches that focus on the sequence of discrete and well-defined steps, like the Waterfall model, where communication channels are realized by passing documents, and others, like the Agile model, which emphasizes on the need for flexibility and constant, direct communication between team members. These newer models are very popular with software teams of varying sizes. Due to the importance and the means of communications described by these models, it is desired to recruit people that possess both technical and communicational skills. The problem, though, that arises when looking for people like these, lies in the difficulty of assessing these skills.

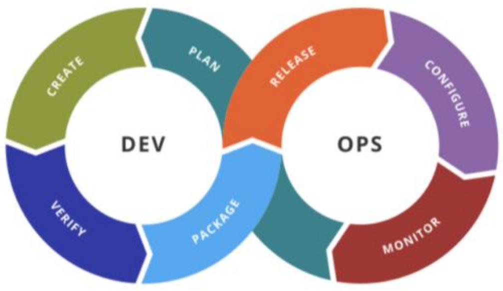
Within the context of this project we focus on this issue. To do so we employ data mining techniques for identifying different team roles and also assess the activities of team members within the software development and operations process. The implemented system draws user activity data from the GitHub web platform and uses them as input to cluster team members. This way we attempt to provide insight into the different team member roles that appear in open source projects, like the ones at GitHub, and the performance of the users that act under these roles.

After extensive experimentation with different combinations of datasets and evaluation features, the results that are presented as final are considered to offer critical insight into those matters.

II. Research Overview

Our dataset consists of features about every contributor engineer in GitHub's 240 most stared projects, which have 5 to 10 contributors. The data was collected from GitHub's API [1]

Our first part of research was focused on engineers' roles classification in every project. The second part was focused on the engineers' quality of contribution evaluation in every role. In order to come on a conclusion about these questions we separated engineers in three main roles Dev, DevOps and Ops. Dev stands for developers who are responsible for the creation and the verification of project's code. DevOps stands for Developer and Operations who are responsible for the releases and the planning process of the project. Ops stands for operations who are responsible for monitoring and configuration.



We used three clustering algorithms (K means, DBSCAN and Hierchical Clustering Complete linkage). The evaluation for the selection of the preffered parameters for each algorithm was done by repeatedly running each algorithm with different parameter values and compare the results based on each algorithms valuation metrics.

You can use this section to possibly analyze the data and state the research questions that you are trying to answer.

III. System Design

A. System Overview

To answer each one of our questions we selected specific metrics of our dataset to determine the role classification and the contribution evaluation for every engineer. In the first case we applied each clustering algorithm and evaluate the results with Cohesion, Separation and Average Silhouette to determine the algorithm we are going to use to continue our research.

B. Data Preprocessing

The final dataset is a result of a number of pre-processing stages in order to eliminate the error percentage in our final model. In each stage we only keep the engineers who are compatible with the stage's scenario. Each stage's dataset is the entrance to the next filter. Therefore, each dataset is a subset of the next one, excluding the original one.

1. ***First stage**: Missing Values removal*
   It's also important to deal with missing/null/inf values in your dataset beforehand. There are many ways to deal with such values, the one we chose to implement is to remove them.
2. ***Second stage**: Exclude 1-day developers*
   In the first stage we eliminate the engineers who has only contributed to the project for one day. The thought behind this filter was that we can make o reasonable conclusion of their role cause of the short time of their activity. The filter was structured based on activity_period_in_days metric of our dataset.
3. **Third stage**: *Exclude short time contributors*
   In this stage we excluded the 10% of the engineers who participated the least in the projects. We ordered the dataset based on activity_period_in_days metric and keep the upper 90% of the dataset.
4. **Fourth stage**: *Data normalization*
   At this final stage of the dataset's process we normalized the scale of metric's values, because each observations' feature values are represented as coordinates in n-dimensional space (n is the number of features) and then the distances between these coordinates are calculated. If these coordinates are not normalized, then it may lead to false results.

This whole process is done to optimize the clustering.

IV. Roles calsification

A. Model Construction

In order to demonstrate the role classification, we used three clustering algorithms DBSCAN, K means and Hierchical Clustering with complete linkage and both Euclidean and Manhattan distances. We based role classification on 4 metrics from the final dataset resulted after the pre-processing act (issues_participated, issues_opened, issues_closed, commits_authored). The results of each algorithm are going to be represented separately in the following sections.

DBSCAN

DBSCAN is a known density-based algorithm used in datasets with different data densities. Therefore, DBSCAN is failing to extract information-rich clusters from our dataset mainly because the data points are being concentrated around zero and categorizes them in the same cluster. Although we can use DBSCAN as an extra stage of preprocessing in order to get rid of some outliers. These outliers were going to form clusters with not enough data, so it would be very difficult to extract conclusions about the role evaluation of each group. Thus, we prefer to focus in our main distribution of data at the beginning of the axes. As the result of this final stage of preprocessing we produce the dataset that we call no irregulars

K means

The problem with K means relies on the fact that our data is not equally distributed. Unfortunately, our data points' density around the axes center are higher than normally expected resulting not well-defined roles. Furthermore, the density of our data around zero creates misleading high evaluation metrics. This conclusion is easily understandable by the results of the original data metrics Fig. 2. The reason that we use only this metric is that as we can see Cohesion (as well as Separation) cannot be compared between the datasets. This is happening because each observations' feature distances which are being examined are extremely different from each and cannot be put in the same unit system.

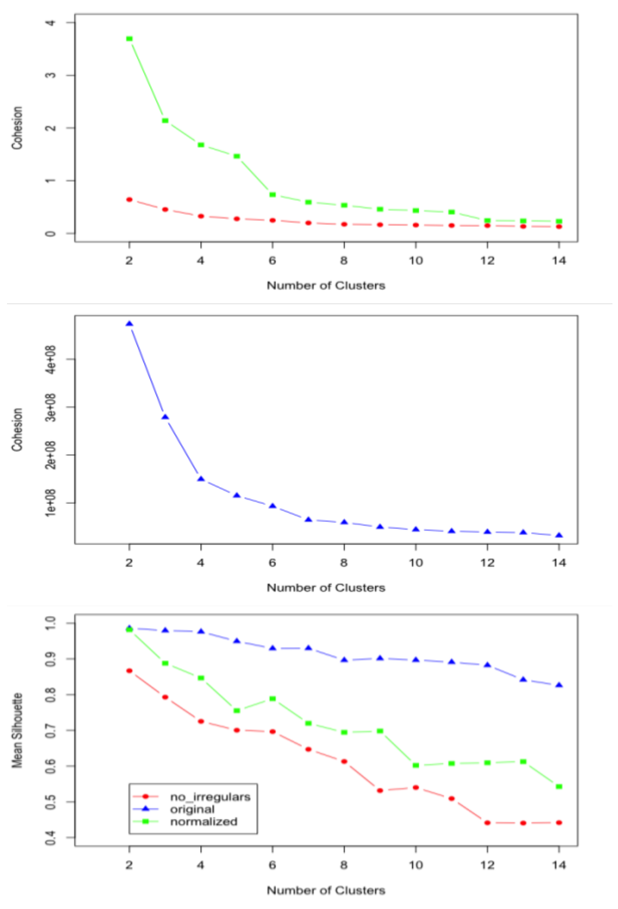
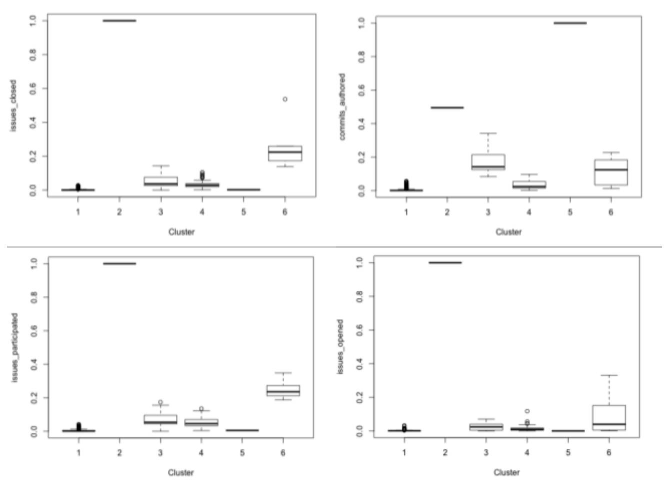


Fig. 1. Role Classification Results with K means.



Model 1. Role Classification Results with K means.

Hierarchical Clustering

Out of the three methods of clustering the one that we choose to continue with is Hierarchical Clustering with complete linkage. As we can see from the figures hierarchical clustering provides the best division between the clusters as well as sufficient results in various metrics. Furthermore, it is the most compatible with our data because it can handle irregular clusters, also we tried to eliminate as much noise points as possible in order to get the best results from it. The only problem with this method is the creation of one cluster where its values concentrate at the beginning of the axes and we are unable to extract enough information. Therefore, we are going to use again hierarchical clustering in order to further examine this particular cluster and identify the roles that it contains. Here we must point out that in this examination the results we get can only be compared with the values inside the cluster and not with the other results of clustering

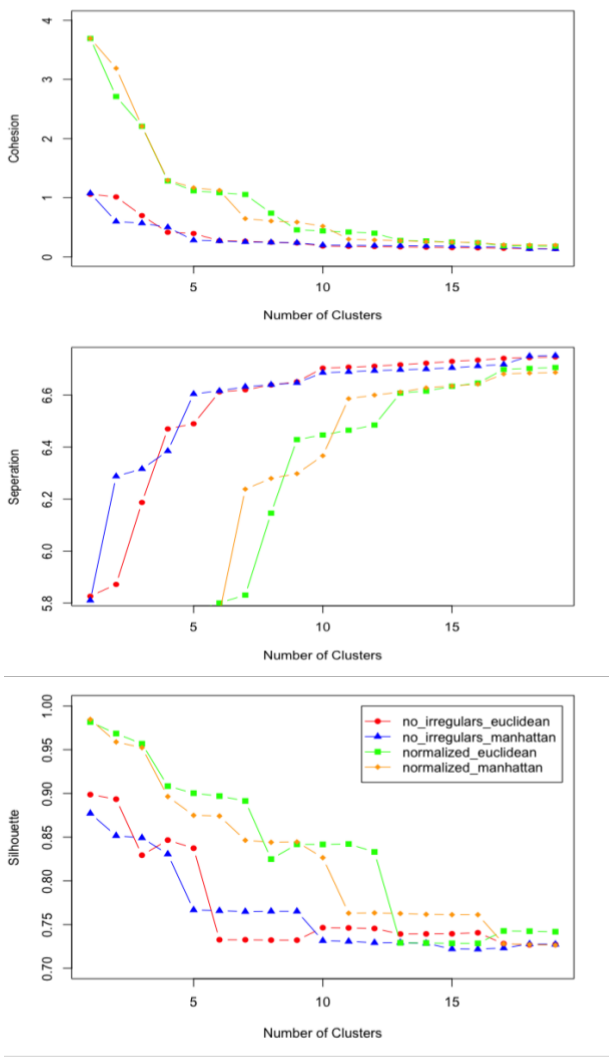
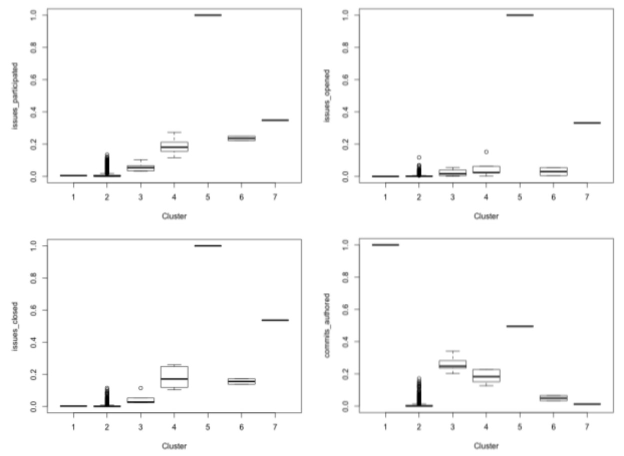


Fig. 2. Role Classification Results with Hierchical Clustering.

A. *Model Evaluation*

As we can see from the figures it is more preferred to use the normalize euclidean dataset. From the elbow method we can use either 6, 7 or 8 clusters without any significant change at the result of the metrics. Although through further examination and visualization of the data we jump to the conclusion that it is best to use 7 clusters for our analysis.



Model 2. Role Classification with Hirerchical Clustering, dataset = normalized, distance = euclidean, linkage = complete, n_clusters = 7

V. Roles Interpretation

In this section we try to identify the specific roles that an engineer can have in a project based on our dataset.

B. *Evaluation Methodology*

The characteristics we examine in order to get our conclusions are ("Issues Opened", "Issues Closed", "Commits Authored" , "Issues Participated"). Thus, we have the following groups

- Dev

  a. *Commits Authored*

  b. *Issues Closed*
- Ops

  a. Issues Participated
- DevOps
  - Issues Opened
  - Issues Closed
  - Commits Authored
  - Issues Participated

In order for a cluster to belong into a specific group its values must be relatively high in those specific categories compared to the other clusters. Although we must point out that this is not the only criterion that needs to be met in order to categorize a cluster. More specifically if a cluster belongs in the DevOps group that does not necessarily means that its values are high in all of the above categories but just a relative good combination of those would be adequate. Also, it is possible that the analysis would result in new categories except the above.

C. *Evaluation Results*

	Issues Participated	Issues Opened	Issues Closed	Comments Authored	Role Hypothesis
1	Low	Low	Low	High	Dev
2	Low	Low	Low	Low	Undecided
3	Medium	Low	Low	High	Dev
4	High	Low	Medium	Medium	DevOps
5	High	High	High	High	DevOps
6	High	Low	Medium	Low	Ops
7	High	High	High	Low	Project Owner

As shown above we categorize the 1st cluster as Dev. It has the highest commits in the dataset and the other characteristics have low values as well as the "Issues Closed" which we said it would be important for the the Dev cluster. That happens because as we know at the Github the privilege to close issues can be granted only from the repository owner. So it is possible that the Dev cluster consists some engineers that contribute to the repositories without being owners or have the privilege to close issues. The 2nd is being categorized as Undecided because it is difficult to jump to any conclusions with all of the characteristics being "Low". Thus we need to further examine this particular cluster and that's because it contains most of our data and we cannot ignore it. Also in the 7th cluster the values of the characteristics are unlike anything that has been seen before and that's the reason we have to categorize it as a new group that we have not predicted before

VI. Deeper Roles Interpretation

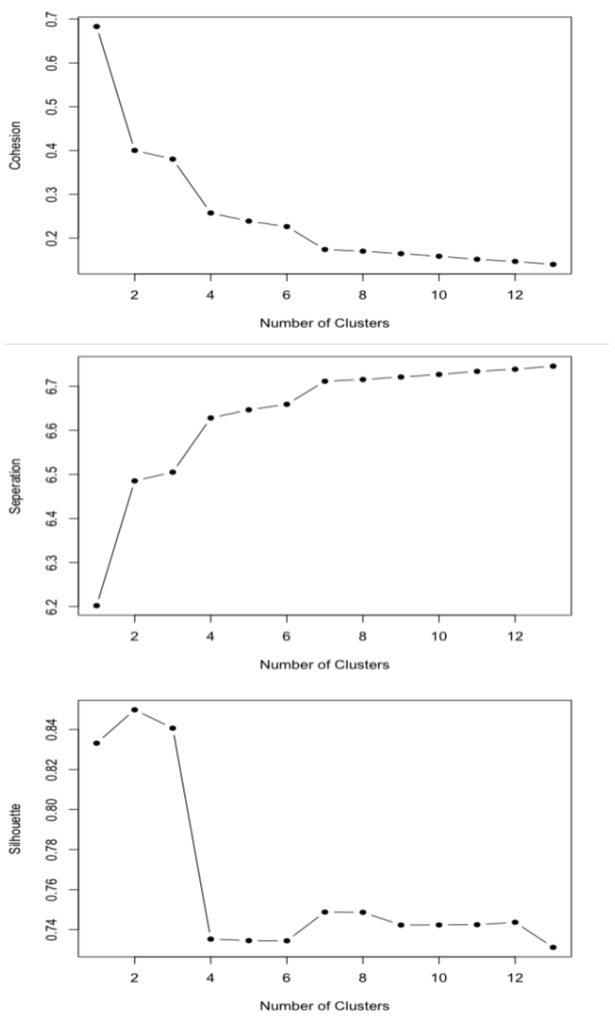
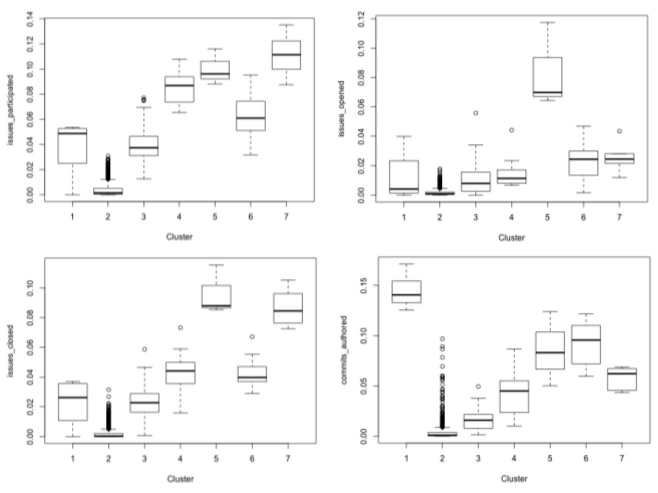


Fig. 3. Deep Role Classification Results with Hierchical Clustering.

Here we produce a specific model in order to further examine the 2nd cluster as has been said. In this model we also use Hierarchical Clustering and we get the metrics of Cohesion Separation and Silhouette as shown in he above figures. Again using the elbow method we choose to continue our analysis with 7 clusters



Model 3. Deeper Role Classification with Hirerchical Clustering, dataset = deep_normalized, distance = euclidean, linkage = complete, n_clusters = 7

A. Evaluation Methodology

Here we follow the same rules as the first model. But it is important to say that the values of this model can only be compared to each other and not with the values of the first model. Therefore the meaning of "Low" "Medium" "High" is different in each model but here too we have the same clustering targets.

D. Evaluation *Results*

parameters	Issues Participated	Issues Opened	Issues Closed	Commits Authored	Role Hypothesis
1	Low	Low	Low	High	Dev
2	Low	Low	Low	Low	Undecided
3	Low	Low	Low	Low	Ops
4	High	Low	Medium	Medium	Ops
5	High	High	High	Medium	DevOps
6	High	Medium	Medium	High	DevOps
7	High	Medium	High	Medium	DevOps

Here the 2nd contains again most of the data but its distribution and most of its values are extremely Low so we can not extract any useful information about how to characterize it. Although the 3rd cluster also contains Low values considered the overall dataset but they are relatively higher than the 2nd cluster and that is why we can categorize it as Ops. In the other clusters we follow the same approach as the first model in order to categorize them.

VII. Roles Evaluation

In this section we will analyze the performance of each engineer in each role. The same methodology will be implemented for this evaluation as we are going to create clusters for each role. The algorithm we will use for this is the Hierchical Cluastering and the dataset will be the normalized euclidean dataset as the concept of this expiriment remains the same with the role classification.

i. Dev Evaluation

In this examination we are not necessarily looking for a large number of separate clusters but we rather prefer a smaller number of clusters which are richer in information. We came to this conclusion because now we are not looking for separate roles of engineers but we try to find commons characteristics between the Devs. In the Dev cluster we include both the Devs of the original analysis and the Devs that came from the further examination of the 2nd cluster. As we can see from the figures which refer to the Dev cluster we choose using the elbow method to have 4 clusters. The main characteristics of those clusters are

1. Productive: If the team has high "Issues Closed Per Day"
2. Responsive: If the team has low "Average Time to Close Issues"
3. Elegant: If the team has low "Violations Added"

ii. DevOps Evaluation

The difference of this evaluation is that we examine a team which has characteristics both from Dev and the Ops cluster. Therefore it would be more useful to have slightly more teams than the other 2 evaluations in order to have more distinct roles. Using the same methods as before and looking at the DevOps figures we come up 8 distinct teams.

iii. Ops Evaluation

Here we are looking for distinct teams that show how much engaged and helpful is an Op in a particular project. Just like before we are using the elbow method in order to find the optimal number of clusters which is 4. Basically we follow the same logic as the Dev Evaluation and we can characterize an Op as:

1. Involved: If the team has low "Inactive Time"
2. Verbal: If the team has high "Average Comments Length"
3. Productive: If the team has high "Average Comments Per Issue"

A. Evaluation Methodology

In order to evaluate the Dev, Ops, DevOps clusters we extract certain metrics from the original

- Dev
  a. Violations Added
  b. Violations Removed
  c. Average Time To Close Issues
  d. Closed Issues per Day
- Ops
  a. Average Comments Per Issue
  b. Average Comments Length
  c. Issues Opened
  d. Inactive Period
- DevOps
  a. Average Comments Per Issue
  b. Average Comments Length
  c. Violations Added
  d. Violations Removed
  e. Average Time To Close Issues

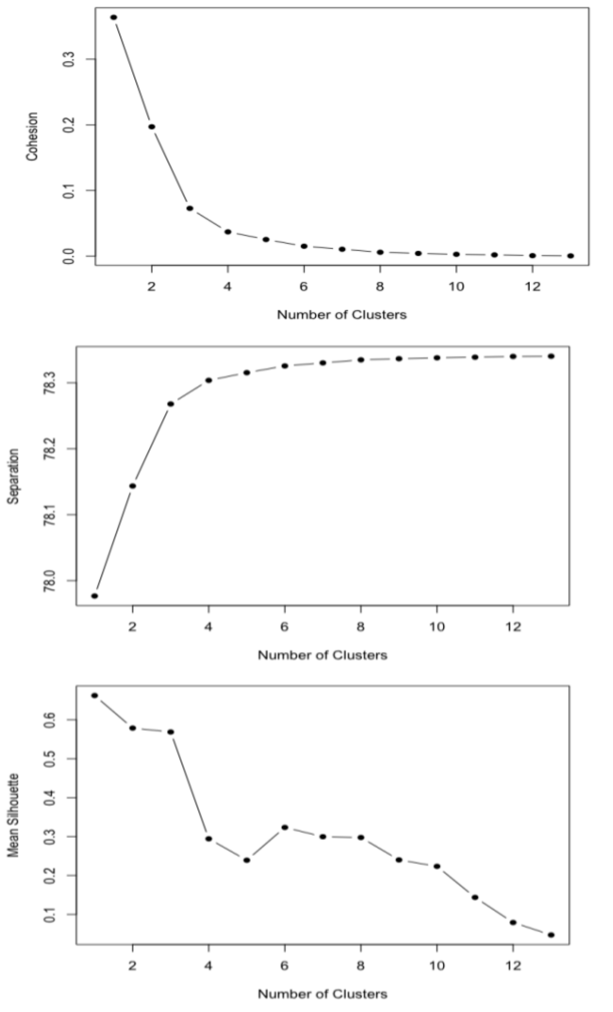
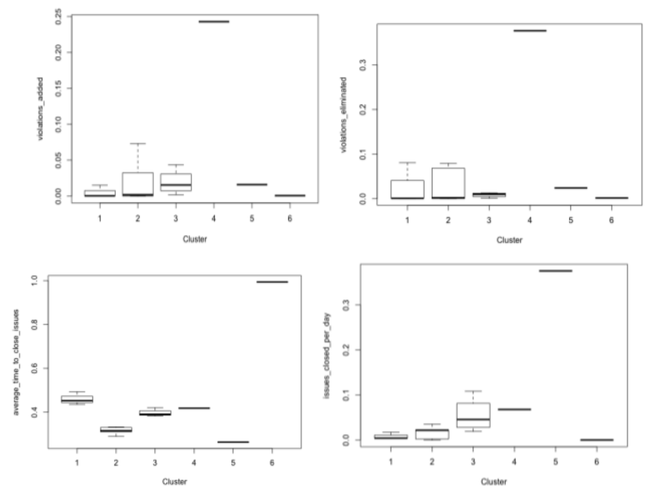


Fig. 4. Dev Evaluation with Hierchical Clusterin, dataset = normalized, distance = euclidean, linkage = complete.



Model 4. Dev Evaluation

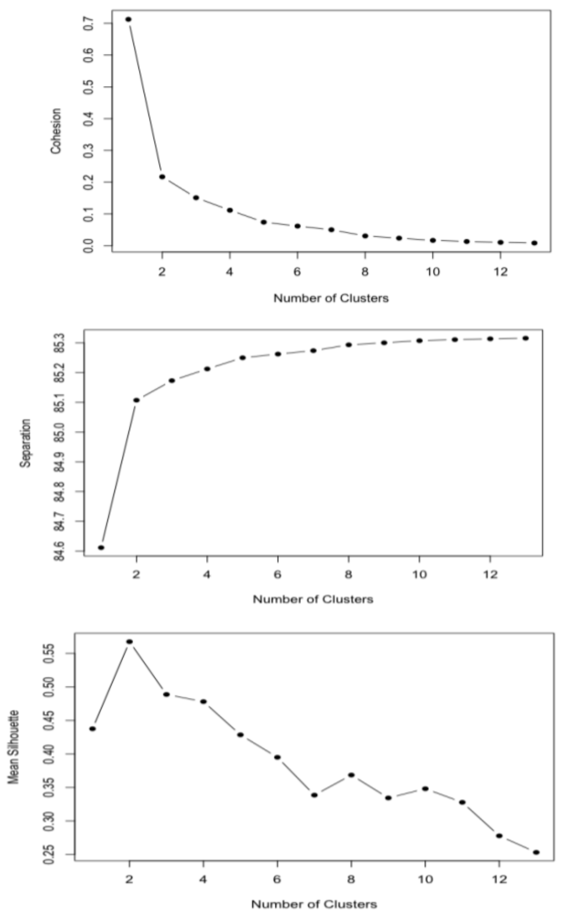
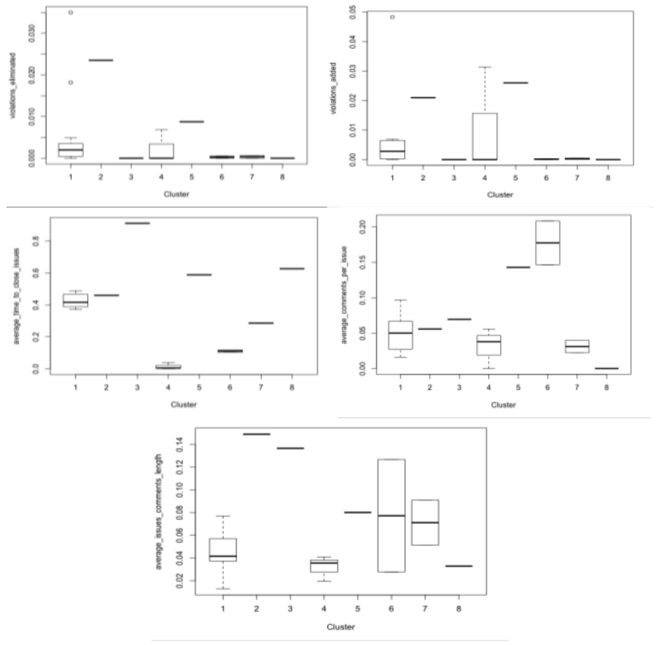


Fig. 5. Dev Evaluation with Hierchical Clusterin, dataset = normalized, distance = euclidean, linkage = complete.



Model 4. Dev Evaluation

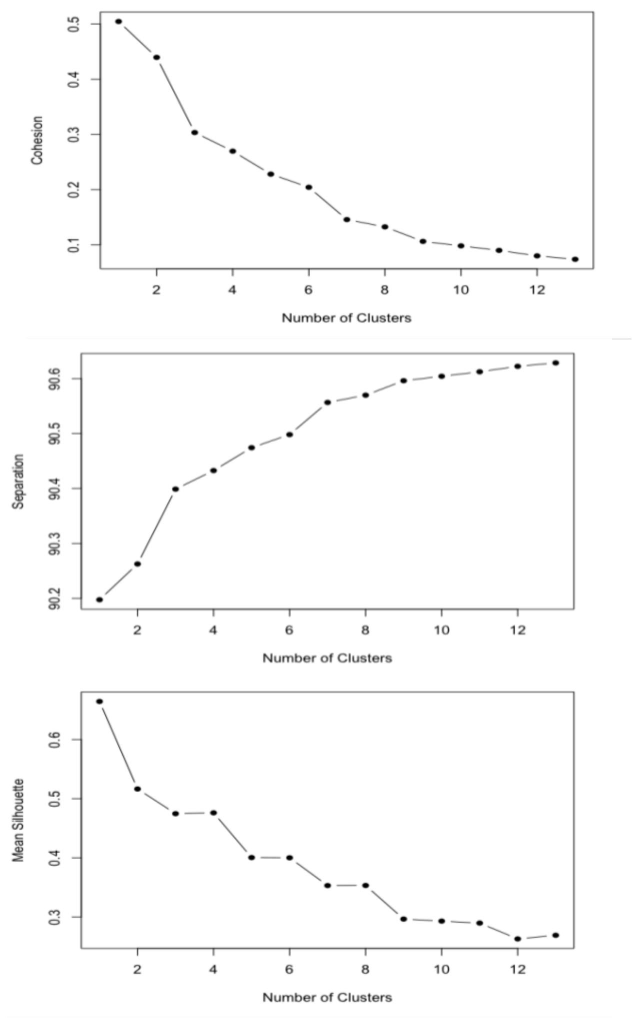
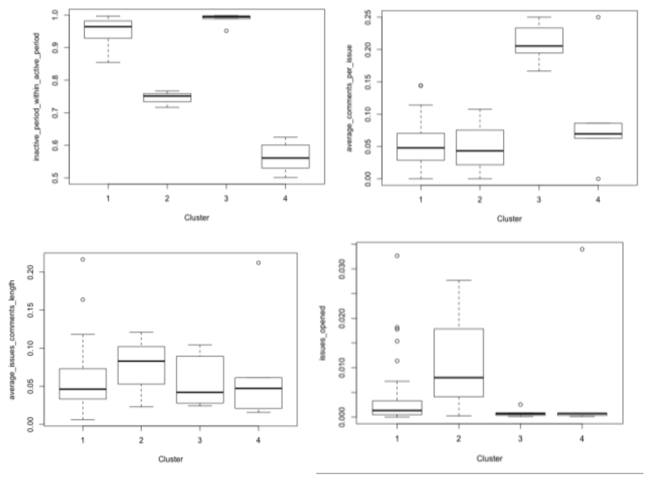


Fig. 6. Ops Evaluation with Hierchical Clustering ,dataset = normalized, distance = euclidean, linkage = complete.



Model 4. Dev Evaluation

B. *Evaluation Results*

In this section we present the results of the above analysis and we categorize each original cluster in teams based on characteristics that we extract from our original dataset

	Violations Added	Violations Removed	Average Time to Close Issue	Issues Closed per Day	Evaluation Hypothesis
1	Low	Low	High	Low	Productive Responsive Elegant
2	Medium	Medium	Low	Low	Productive Responsive Elegant
3	Medium	Medium	Medium	High	Productive Responsive Elegant
4	High	High	High	Medium	Productive Responsive Elegant
5	Low	Low	Low	High	Productive Responsive Elegant
6	Low	Low	High	Low	Productive Responsive Elegant

Table 1. Dev Evaluation Model

DEV OPS	Violations Removed	Violations Added	Average Time to Close Issue	Average Comments per Issue	Average Comments Length	Evaluation Hypothesis
1	Medium	Medium	Medium	Medium	Medium	Responsive Elegant Involved Verbal
2	High	High	Medium	Medium	High	Responsive Elegant Involved Verbal
3	Low	Low	High	Medium	High	Responsive Elegant Involved Verbal
4	Medium	Medium	Low	Low	Low	Responsive Elegant Involved Verbal
5	Medium	High	High	High	Medium	Responsive Elegant Involved Verbal
6	Low	Low	Low	High	Medium	Responsive Elegant Involved Verbal
7	Low	Low	Medium	Low	Medium	Responsive Elegant Involved Verbal
8	Low	Low	High	Low	Low	Responsive Elegant Involved Verbal

Table 2. DevOps Evaluation Model

	Inactive Time	Average Comments per Issue	Average Comments Length	Issues Opened	Evaluation Hypothesis
1	Low	Low	High	Low	Involved Verbal Productive
2	Medium	Medium	Low	Low	Involved Verbal Productive
3	Medium	Medium	Medium	High	Involved Verbal Productive
4	High	High	High	Medium	Involved Verbal Productive
5	Low	Low	Low	High	Involved Verbal Productive
6	Low	Low	High	Low	Involved Verbal Productive

Table 2. DevOps Evaluation Model

VIII. Conclusions

The initial target of this examination was to successfully identify the different roles that engineers can have in a project. And then evaluate each role based on some characteristics. It’s obvious that when it comes to a software project it is rather difficult to evaluate someone who participates in it. So it goes without saying that in some cases we simplified the data and the models which are being used for complexity reasons and only if it was necessary we have made a more thorough examination. In this case we considered it would be more sufficient to use 4 different models in order to choose the most appropriate one for our data. Also we get 3 different datasets from the different steges of preprocessing and

As a final observation we must point out that most of the characteristics that has been used in the role interpretation were oriented towards the Dev and not the Ops. This is happening because GitHub as a platform appeals more to the Devs and not the Ops and the dataset contained very few PureOps characteristics. Furthermore, most of the data were concentrated at the begging of the axes around zero. So, it was difficult to identify distinct clusters with enough data points to get thorough results. Although we consider that we have achieved our target of successfully finding the different roles of engineers though our method of preprocessing and the most compatible clustering with our data

References

[1] “GitHubAPI”[Lastaccessed:5December2018],Availableat:https://developer.github.com/v3/

[2] I. Zafeiriou, "Software engineer profile recognition through application of data mining techniques on GitHub repository source code and comments," *Diploma thesis*, Aristotle University of Thessaloniki, Thessaloniki, Greece, 2017.