ANALYZING SIMILAR BUILD CONFIGURATIONS ON GITHUB

1.INTRODUCTION

GitHub is the home of hundreds of millions of OSS projects, fostering collaboration among millions of users to create innovative software projects. However, as the number of projects grows, so does the challenge for developers to effectively manage and optimize their workflows, prompting the need for automated solutions. While there have been previous attempts at finding similar GitHub projects, the build configurations have been overlooked until now.

2.OBJECTIVE

In this research, we are answering questions such as:

- Is OSS similarity analysis feasible from the perspective of build configurations?
- How can we measure project similarity in terms of Maven build plugins?
- Are there any patterns to be uncovered by analyzing workflow files?
- How does build configuration analysis differ from other well-established frameworks?

AUTHORS

Calin-Mihnea Manoli
c.m.manoli@student.tudelft.nl

AFFILIATIONS

Delft University of Technology

TU Delft

3.METHODOLOGY

Various research methods have been employed throughout this research, such as:

1. **GitHub Repository Selection:**
   - 2 datasets selected: Maven (743 projects) and Workflow (846 projects).
   - Projects chosen based on queries: "path:*pom.xml" and "path:.github/actions/**.y*ml".
   - Additionally, 2 validation datasets extracted from the CrossSim project [2].
2. **File Retrieval and Preprocessing**:
   - Extracted pom.xml and pipeline files.
   - Additionally, for Maven analysis:
     - Used Maven tools to build the project and extracted the effective-pom.xml file.
     - Constructed a table containing the projects and all possible plugins + Java versions.
3. **Similarity Metrics**:
   - Applied Term Frequency - Inverse Document Frequency (TF-IDF) [4] vectorization technique:
     - Maven analysis: done based on the project-plugin table.
     - Workflow analysis: encode files individually.
   - Computed the cosine similarity between each vector and stored the results in a similarity matrix.
   - Applied Principal Component Analysis (PCA) to reduce the dimensionality to 2.
   - Applied K-means to cluster the data based on their similarity.
4. **Comparative Analysis**:
   - Compared our results to the RepoPal [1] using the validation sets.

4. EXPERIMENTS AND RESULTS

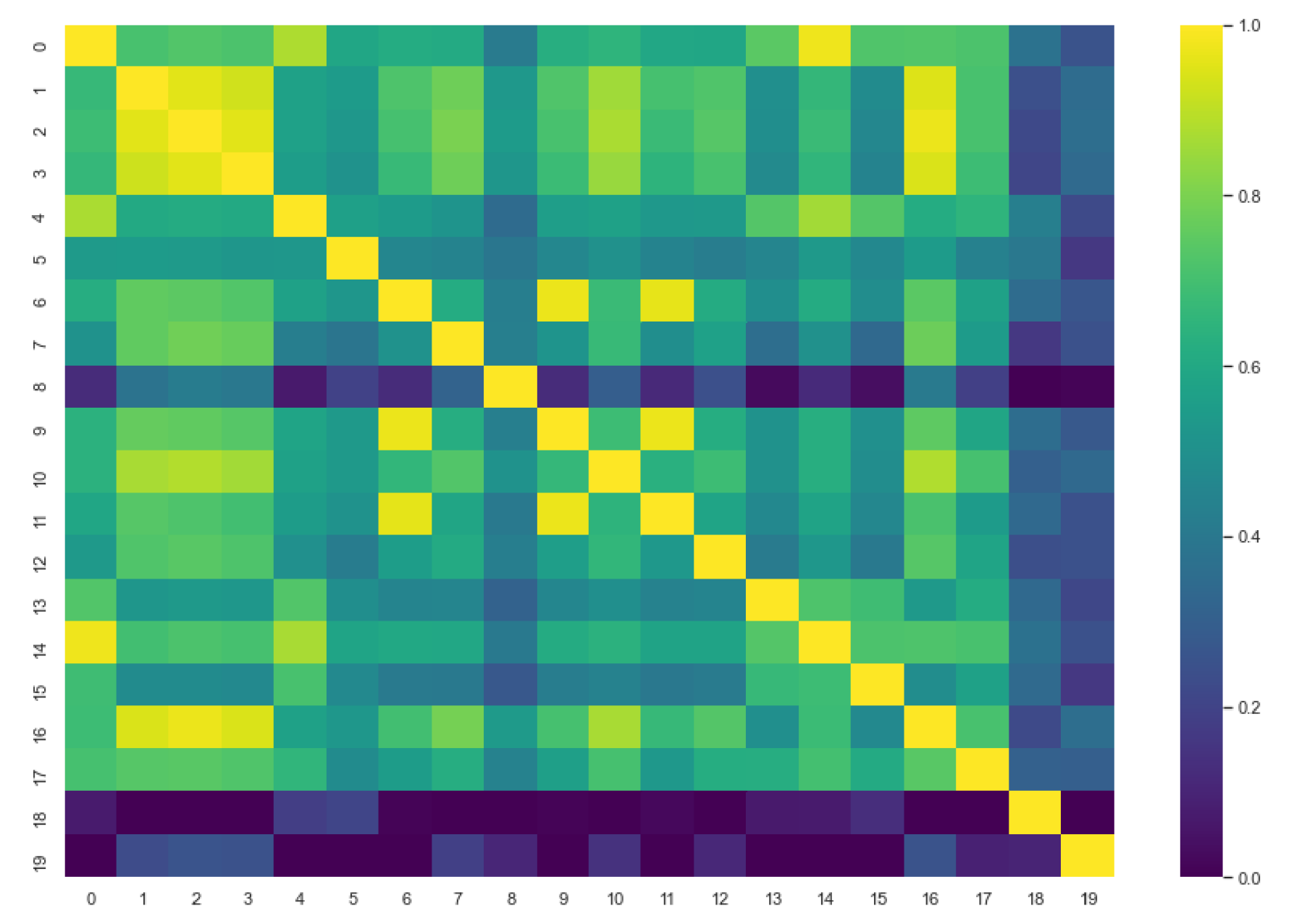


Figure 1: The cosine similarity matrix produced for the first 20 Maven projects within the dataset.

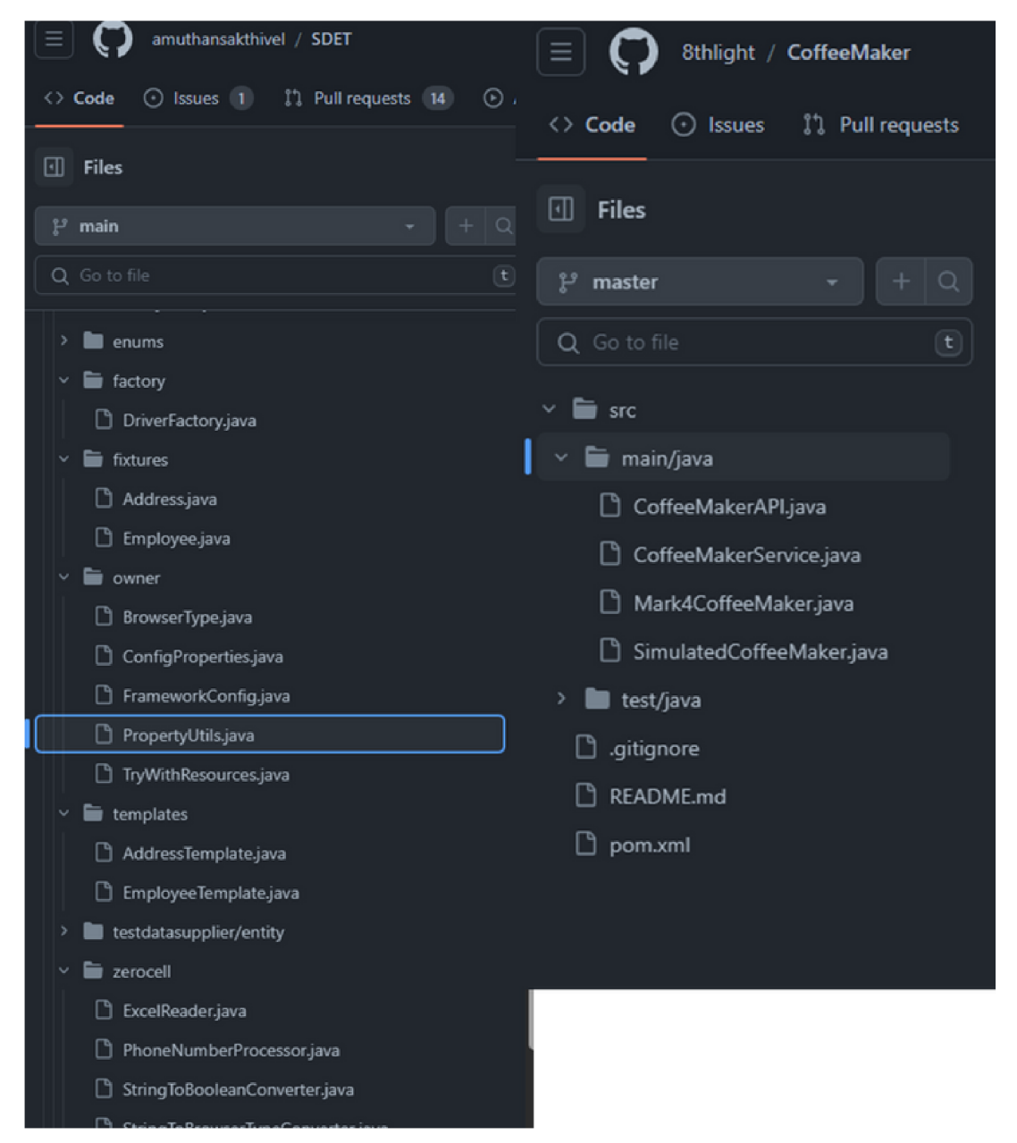


Figure 3: Side-by-side comparison of seemingly different projects.

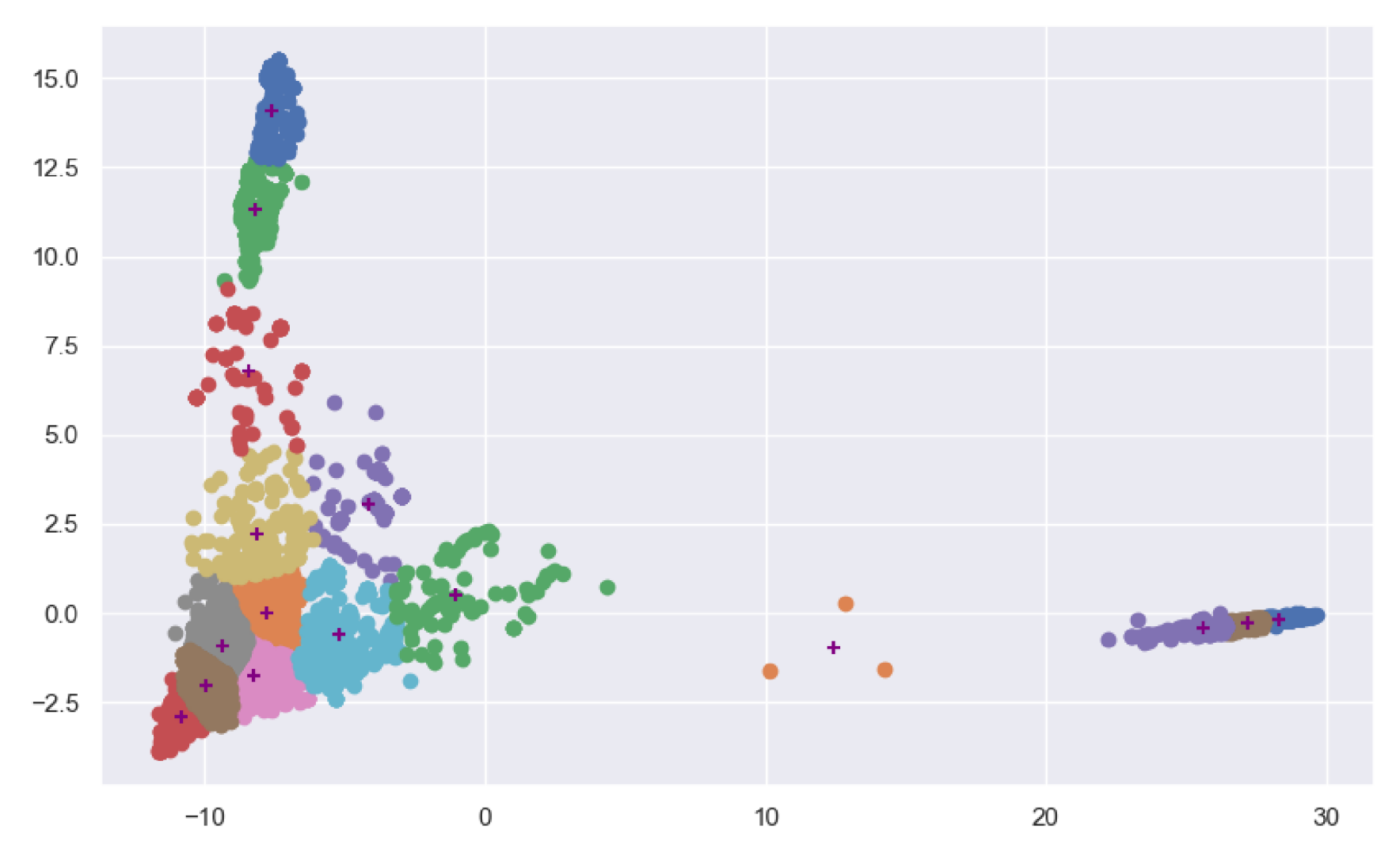


Figure 5: The resulting 16 clusters after applying PCA and K-Means clustering on the Workflow projects.

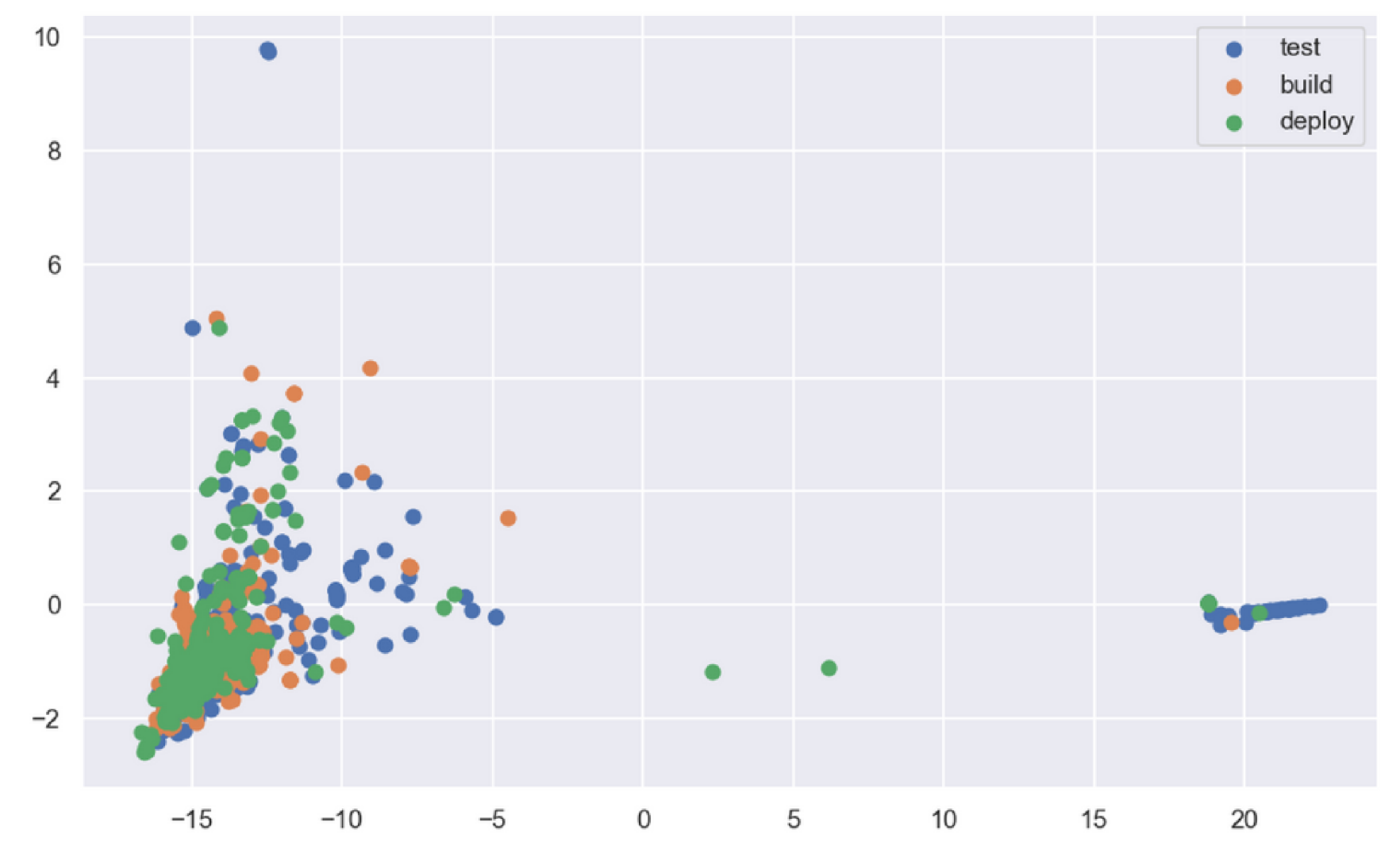


Figure 6: Workflow projects clustered by file category instead of K-Means clustering. Only 3 categories are displayed to facilitate

- Clustering based contents uncorrelated with clustering based on category (Adjusted Rand Index = 0.03).
- Average file similarity for same organisation:0.56.
- Average similarity for randomly selected files: 0.17.
- Spearman correlation with

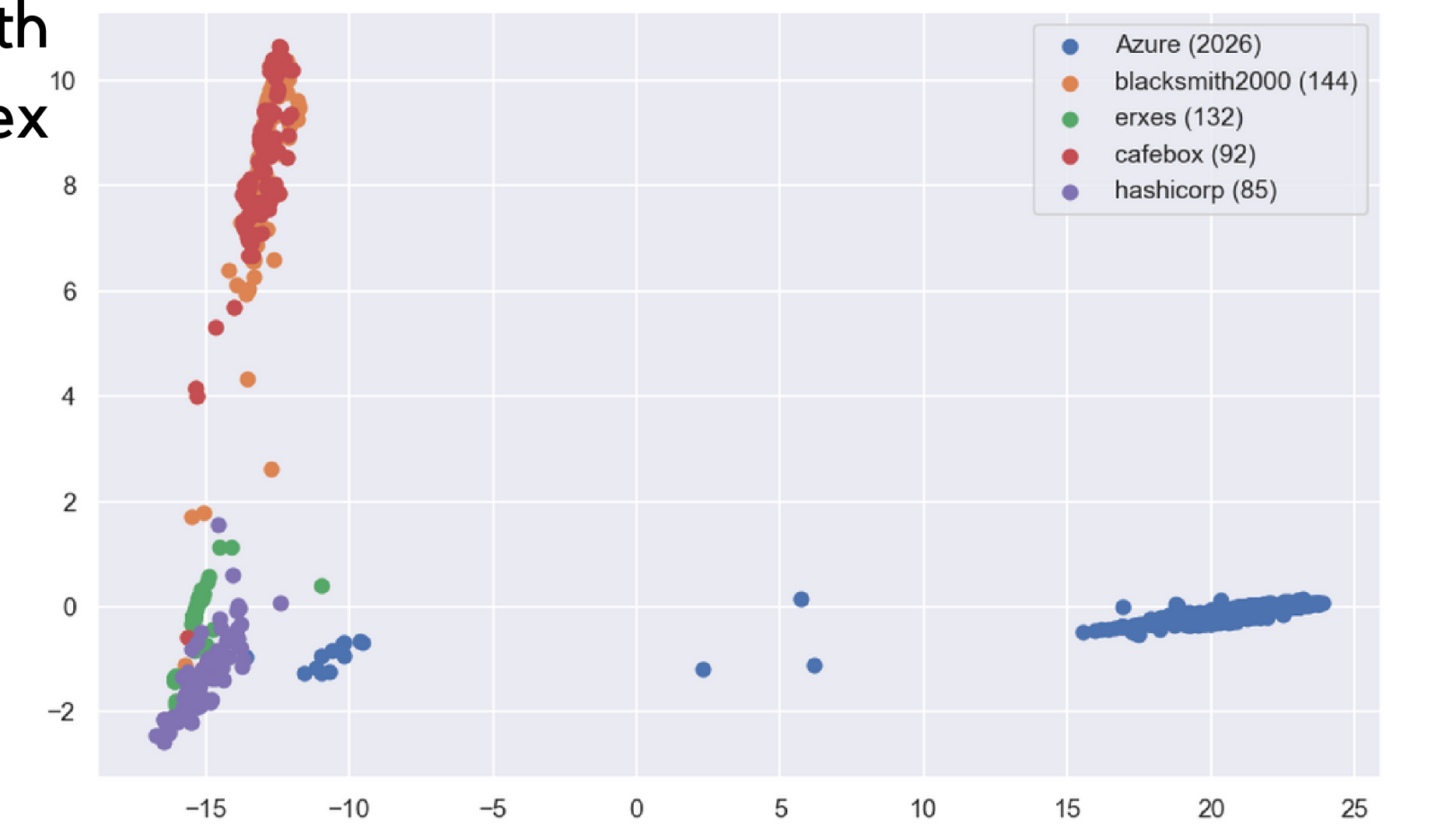


Figure 7: Workflow projects clustered by the responsible organization, displaying the top-5 organizations in terms of the number of files created.

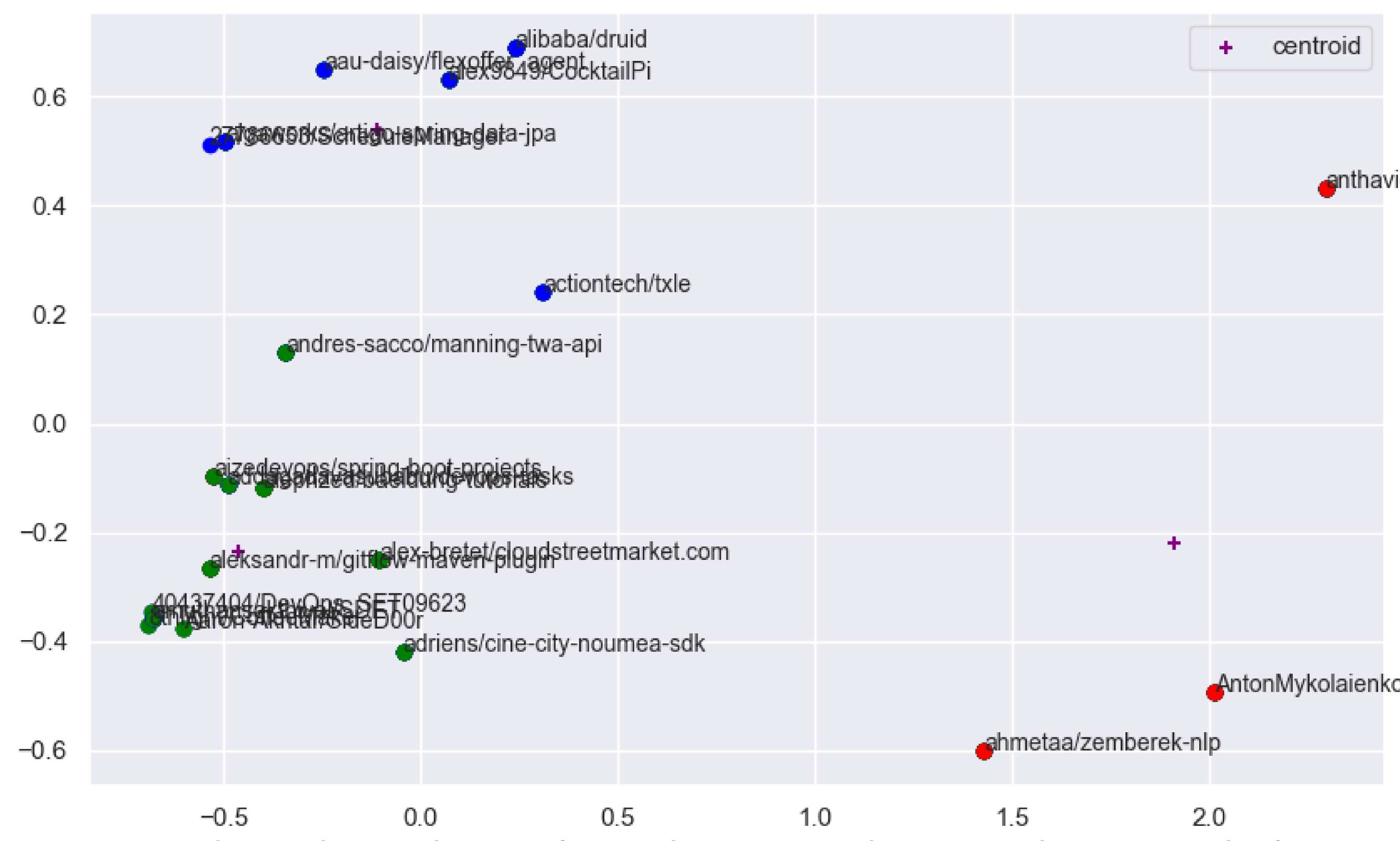
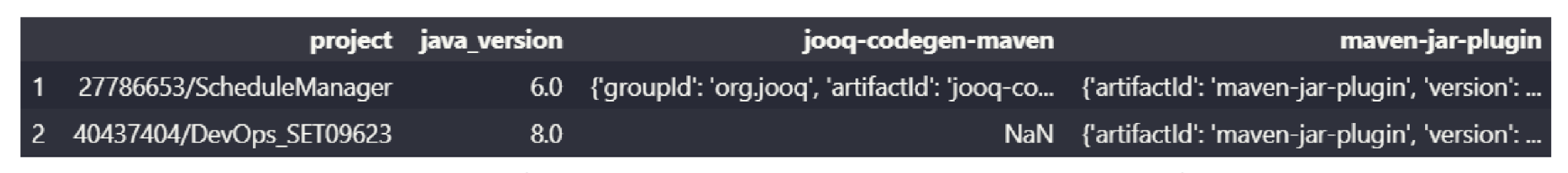


Figure 2: The resulting 3 clusters after applying PCA and K-Means clustering on the first 20 Maven projects.

- Maven analysis is feasible
- Spearman correlation with existing tools: 0.033



	project	java_version	jooq-codegen-maven	maven-jar-plugin
1	27786653/ScheduleManager	6.0	{'groupId': 'org.jooq', 'artifactId': 'jooq-co...	{'artifactId': 'maven-jar-plugin', 'version': ...
2	40437404/DevOps_SET09623	8.0	NaN	{'artifactId': 'maven-jar-plugin', 'version': ...

Figure 4: An example of the project-plugin table, built for only 2 projects.

5.FUTURE WORK

- Expand dataset diversity by incorporating projects from various platforms and considering additional build configurations like Gradle, Docker, Kubernetes, and Makefiles.
- Augment data collection with additional facets such as file update frequencies and pipeline runtimes for a more nuanced analysis.
- Apply stemming techniques to enhance accuracy by reducing TF-IDF vector space.
- Develop practical applications, such as recommender systems, to translate gained insights into actionable tools for developers, aiding real-time project configuration choices.
- Integrate results with code, documentation, and dependency similarity for comprehensive project understanding.

6.CONCLUSION

Our study not only demonstrates the feasibility of analyzing project similarity through build configurations on GitHub but also emphasizes the potential for discovering hidden patterns and connections among projects with diverse goals and functionalities. This approach challenges traditional notions of project similarity, uncovering meaningful insights beyond source code analysis.

RELATED LITERATURE

[1] Yun Zhang, David Lo, Pavneet Singh Kochhar, Xin Xia, Quanlai Li, and Jianling Sun. Detecting similar repositories on github.
[2] Phuong T. Nguyen, Juri Di Rocco, Riccardo Rubei, and Davide Di Ruscio. An automated approach to assess the similarity of github repositories.
[3] S. Kawaguchi, P. K. Garg, M. Matsushita and K. Inoue. MUDABlue: an automatic categorization system for open source repositories.
[4] Luḱ as Havrlant and Vladik Kreinovich. A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation).