

Lab Assignment 4: Exploring cyclomatic complexity (MCC) changes in Open-Source Repositories

INTRODUCTION, SETUP, AND TOOLS

Introduction

The purpose of this lab is to analyze the impact of code changes (git diff) on McCabe's cyclomatic complexity metric (MCC) in OSS ecosystems like GitHub. we need to analyze cyclomatic complexity values before and after changes in code files. and lastly the impact of source code changes (git diff --histogram) on the well-known McCabe's cyclomatic complexity metric. we used **Pydriller** to extract commit data, **Lizard** to calculate MCC, and **Py2cfg** to generate Control Flow Graphs (CFGs)

Setup

we need to have Pydriller Lizard Py2cfg Matplotlib

- **Installation:** we need to install **pydriller**, **Lizard**, **Py2cfg**, and **Matplotlib** on the local machine before starting lab.
- **Python Installation:** We need to have Python 3.10.

Tools

- **pydriller:** is an open-source Python library that allows you to "drill into" git repositories.
- **Lizard Installation:** used for calculating cyclomatic complexity.
- **Py2cfg Installation:** is used for generating control flow graphs (CFGs) from Python code.
- **Matplotlib:** is a comprehensive library for creating static, animated, and interactive visualizations in Python.

METHODOLOGY AND EXECUTION WITH RESULT AND ANALYSIS

Repository Selection:

For the analysis I selected the repository [sensAI](#) [2]. This is an AI-based project that focuses on providing solutions for sensor data analysis, leveraging machine learning techniques for predictive analytics and anomaly detection. It fits the criteria of a medium-to-large scale open-source repository.

Define Selection Criteria:

As this aligns with your previous lab, and we like to keep it same criteria for this analysis as well to establish where I use the SEART GitHub Search Engine [3] and established the following **selection criteria**:

Search: In search I added AI so that I get repository which are working either with AI or on AI.

Language: In Language I added python so that I get repository which are working either with python or on python as recommended.

Number of Commits: The repository must have commits between **1000 and 1200**. This will indicates that the repository is actively developed and also not too large to make analysis difficult.

And selected [sensAI](#) [2] repository as first preference. the following image shows the process and output.

The image shows two screenshots of the SEART GitHub Search Engine interface. The top screenshot displays the search filters, and the bottom screenshot displays the search results.

Search Filters:

- General:** AI (Contains), Python (Uses Label)
- History and Activity:** Number of Commits (1000 to 1200), Number of Contributors (min to max), Number of Issues (min to max), Number of Pull Requests (min to max), Number of Branches (min to max), Number of Releases (min to max), Number of Stars (min to max), Number of Watchers (min to max), Number of Forks (min to max), Size of codebase (Non Blank Lines, min to max), Code Lines (min to max), Comment Lines (min to max).
- Date-based Filters:** Created Between (dd / mm / yyyy), Last Commit Between (dd / mm / yyyy).
- Popularity Filters:** Number of Stars (min to max), Number of Watchers (min to max), Number of Forks (min to max).
- Additional Filters:** Sorting (Name, Ascending), Repository Characteristics (Exclude Forks, Only Forks, Has License, Has Open Issues, Has Wiki, Has Pull Requests).

Results: 198

The results are displayed in a table format. The first result is **aai-institute/sensAI** and the second result is **Acly/krita-ai-diffusion**.

Repository	Commits	Watchers	Stars	Forks	Total Issues	Total Pull Reqs	Branches	Contributors	Open Issues	Open Pull Reqs	Releases	Size	Created	Updated	Last Push	Last Commit
aai-institute/sensAI	1137	7	34	3	52	37	19	5	13	0	25	7.74 KB	2020-02-18	2024-07-11	2024-07-11	2024-07-11
Acly/krita-ai-diffusion	1145	67	7799	391	1208	81	35	31	71	2	50	29.31 KB	2023-09-01	2025-02-04	2025-02-04	2025-02-04

Software Tool Setup:

Download, install, and configure pydriller using command given on **Pydriller** [1]

```
pip install pydriller
pip install py2cfg
```

Every thing is same as in Assignment 3 just added new helper function for and removed extra helper function

Compare get old_mcc and new_mcc in Myers' diff and histogram: This function old cyclomatic complexity from the older source_code_before and new cyclomatic complexity column from the source_code

```
def get_cyclomatic_complexity(source_code):
    try:
        analysis = lizard.analyze_file.analyze_source_code("temp.py", source_code)
        return sum(func.cyclomatic_complexity for func in analysis.function_list)
    except Exception as e:
        print(f"Error computing complexity: {e}")
        return None
```

some minute changes in calling commits from **pydriller**

```
for commit in Repository(repo_url, order="reverse").traverse_commits():
    if len(commit.parents) > 1: # Ignore merge commits
        continue

    if commit_count >= 500:
        break

    for modified_file in commit.modified_files:
        diff_histogram = modified_file.diff_parsed
        old_mcc, new_mcc = None, None

        # Compute cyclomatic complexity for old and new versions
        if modified_file.source_code_before:
            old_mcc = get_cyclomatic_complexity(modified_file.source_code_before)
        if modified_file.source_code:
            new_mcc = get_cyclomatic_complexity(modified_file.source_code)

        writer.writerow([
            modified_file.old_path, modified_file.new_path,
            commit.hash, commit.parents[0] if commit.parents else 'N/A',
            commit.msg, diff_histogram, old_mcc, new_mcc
        ])
```

full code with csv: <https://drive.google.com/file/d/18oggG9TQcEBoWxwcyi58TIYVWf8w2d7F/view?usp=sharing> following is screenshot after running this part

After running this Python script, the analysis generates a CSV file (vsp_analysisLab4.csv) which includes the following columns (similar like past just removed the code that was not been used and added new for adding old_mcc and new_mcc):

- `old_file_path`: The file path of the original version.
- `new_file_path`: The file path of the modified version.
- `commit_sha`: The SHA hash of the commit.
- `parent_commit_sha`: The SHA hash of the parent commit (if available).
- `commit_message`: The message describing the commit.
- `diff_myers`: The diff output generated using the Myers algorithm.
- `diff_histogram`: The diff output generated using the Histogram algorithm.

```

1 from pydriller import Repository
2 import csv
3 import lizard
4
5 repo_url = "https://github.com/opcodes81/sensai"
6
7 output_file = "vsp_analysisLab4.csv"
8
9 def get_cyclomatic_complexity(source_code):
10     try:
11         analysis = lizard.analyze_file.analyze_source_code("temp.py", source_code)
12         return sum(func.cyclomatic_complexity for func in analysis.function_list)
13     except Exception as e:
14         print(f"Error computing complexity: {e}")
15         return None
16
17 with open(output_file, "w", newline="", encoding="utf-8") as f:
18     writer = csv.writer(f)
19     writer.writerow([
20         "old_file_path", "new_file_path", "commit_sha",
21         "parent_commit_sha", "commit_message", "diff_histogram",
22         "old_file_MCC", "new_file_MCC"
23     ])
24
25 commit_count = 0
26
27 # Run the analysis on the repository
28 repo = Repository(repo_url)
29 for commit in repo.iter_commits():
30     commit_count += 1
31     # Get the diff for the commit
32     diff = repo.diff(commit)
33     # Get the diff histogram
34     histogram = diff.get_histogram()
35     # Get the diff Myers
36     myers = diff.get_myers()
37     # Write the results to the CSV file
38     writer.writerow([
39         commit.parents[0].hexsha, commit.hexsha, commit.message,
40         commit.parents[0].hexsha, commit.hexsha, commit.message,
41         histogram, myers
42     ])
43
44 print(f"Analysis complete. Results saved in {output_file}")
45
46 # Print the commit count
47 print(f"Commit count: {commit_count}")

```

```

student@vlsi:~/Desktop/STT_LABS/VSP$ /bin/python3 /home/student/Desktop/STT_LABS/VSP/vsp_analysisLab4.py
saved in vsp_analysisLab4.csv
student@vlsi:~/Desktop/STT_LABS/VSP$

```

Analyze trends:

Identify and report the top 3 frequently changed source code files.

```

import pandas as pd

df = pd.read_csv('vsp_analysisLab4.csv')

file_changes = pd.concat([df['old_file_path'], df['new_file_path']]).value_counts()

top_3_files = file_changes.head(3)
print(top_3_files)

```

```

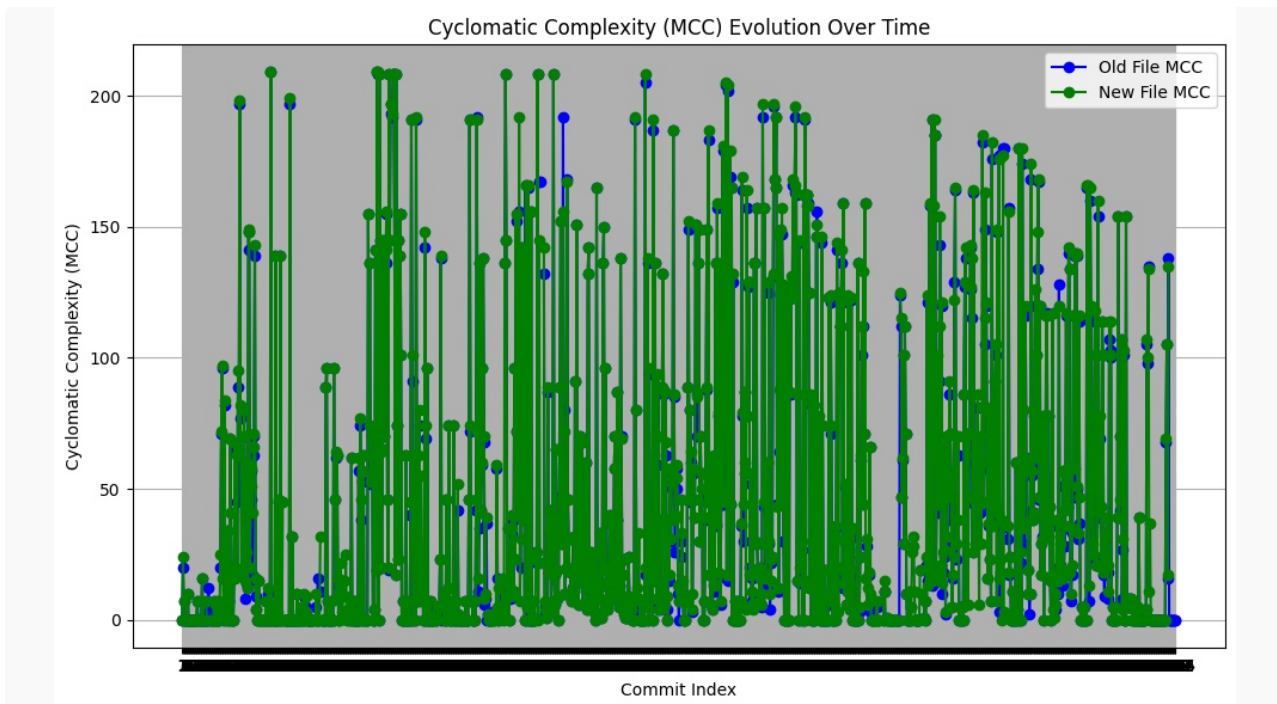
setup.py          96
src/sensai/evaluation/eval_util.py  96
src/sensai/__init__.py  80
Name: count, dtype: int64

```

- **setup.py (96 changes)**
 - This file is usually responsible for package configuration and dependency management.
- **src/sensai/evaluation/eval_util.py (96 changes)**
 - This file is part of the sensai package contains utility functions for evaluation.
- **src/sensai/__init__.py (80 changes)**

- The **init.py** file is used for module initialization.

Plot the changes of cyclomatic complexity values along the timeline of software evolution.



Python code used to generating plotings is in https://drive.google.com/file/d/1bGZt2waLlg1oHMmMbvH_epBwWcPnpVPW/view?usp=sharing

The plot suggests that the cyclomatic complexity of this has increased been constant over time.

More fluctuations in MCC indicate growing complexity in the project.

The blue dots (old file MCC) are generally lower than the green dots (new file MCC), suggesting that changes to the code often introduce more complexity.

The vertical green lines indicate that MCC frequently spikes after modifications.

DISCUSSION AND CONCLUSION

Discussion:

In this lab, we analyzed how code changes in the **sensAI** repository impacted McCabe's cyclomatic complexity (MCC). I found that changes often led to an **increase in Complexity**, reflecting the natural complexity introduced when adding new features or logic. The plot revealed that the **new MCC values** (green dots) were generally higher than the old ones (blue dots), indicating that changes added more decision points to the code.

We also identified key files with frequent changes, such as [setup.py](#), [eval_util.py](#), and [init.py](#), which likely contributed to these increases.

Conclusion:

This analysis shows that software modifications tend to increase cyclomatic complexity, which may suggest growing code complexity and potential challenges in maintaining the codebase. Tracking MCC over time helps developers assess code evolution and identify areas needing attention to prevent excessive complexity. Future work could explore additional quality metrics to better understand the relationship between code changes and software health.

APPENDIX

I would like to express my sincere gratitude to my course instructor, Prof. [Shouvick Mondal](#), for his invaluable guidance and support throughout this lab. I also appreciate the assistance from All TAs, whose help with me with troubleshooting.

Additionally, I am grateful for the resources provided, including Which I have sited down which helped me resolve issues efficiently. Finally, I'd like to thank my peers for contributing to a collaborative and supportive learning environment.

CITATION

[1] ClassroomCode / py2cfg · GitLab. (n.d.). GitLab. <https://gitlab.com/classroomcode/py2cfg>

[2] Ishepard. (n.d.). GitHub - ishepard/pydriller: Python Framework to analyse Git repositories. GitHub. <https://github.com/ishepard/pydriller>

[3] Terryyin. (n.d.). GitHub - terryyin/lizard: A simple code complexity analyser without caring about the C/C++ header files or Java imports, supports most of the popular languages. GitHub. <https://github.com/terryyin/lizard>

[4] Class Slides L3 - <https://drive.google.com/file/d/1zGLtjHGxbytlzDX8gzOai5CSv3ggzxMU/view>

[5] Class Slides L4 - https://drive.google.com/file/d/1gFJe5qDeloQDIEyRdLswWEr_n9xNaH7r/view

[6] Lab Manual - <https://drive.google.com/file/d/1PuUM5NwfCPJOvC1g6cv5131QVOkxUDtg/view>

[7] SEART (Github Repo Search)- <https://seart-ghs.si.usi.ch/>