A comparative analysis of coding approaches in machine learning among computer science and non-computer science students

Grgur Dujmovic
Email: g.dujmovic@student.tudelft.nl / grgur.duj@gmail.com
Responsible professor and supervisor: Gosia Migut

1 Background

The field of Machine Learning is rapidly expanding across disciplines, which is creating demand to teach it to people with diverse educational backgrounds

Computer Science students naturally engage with Machine Learning, but teaching Machine Learning to non-Computer Science students

Previous research has not focused on the difference in approach between CS and non-CS students but rather:

Discovering preconceptions and learning barriers of non-CS students

Exploring different approaches and technologies to use when teaching ML

Teaching ML concepts is challenging due to diverse student programming backgrounds.

This research aims to compare code solutions of two student groups to provide insights that may help in improving Machine Learning education strategies

Research Question:

How do coding approaches for solving Machine Learning problems of computer science students differ compared to students without a computer science background?

Sub-questions:

- Is there a statistically significant difference in code metrics in the solutions of computer science students versus students with no computer science background?
- Can the coding practices of computer science students in machine learning be reliably distinguished by using coding metrics?
- Are there observable differences in comments and readability, between the machine learning code produced by computer science students and those without a computer science background?

2 Methodology

The data is provided and anonymized by TU Delft, the CS students' assignments are from a bonus assignment they completed during their ML course, while the non-CS students' assignments are from a mandatory assignment during a minor

A program is developed that automatically analyzes the code using the following metrics

1. Lines of code
2. Number of functions
3. Number of function calls
4. Number of attributes
5. McCabe's cyclomatic complexity
6. Percentage of comments
7. Number of code cells
8. Number of markdown cells
9. Mean lines per code cell
10. Mean words per markdown cell

These metrics are gathered from previous research and are mostly used for large projects, but they were adapted for smaller projects, focusing on code quality and computational thinking skills.

Common methods: Lines of code, number of functions, number of attributes.

Measures of quality: Cyclomatic complexity, comment percentage.

Computational thinking: Number of code cells, number of markdown cells, mean lines per code cell, mean words per markdown cell, number of markdown words, number of function calls.

A statistical analysis on the collected metrics is performed to determine correlation and statistical significance using confidence intervals, t-tests and Cohen's d

3 Results

The graph below shows bar graphs for each metric with a confidence intervals of 95%. A closer look at each metrics is required to get an overall understanding

Comment Percentage: Non-CS students consistently have a higher comment percentage, but the difference is not significant.

Mean Lines in Code Cells: Both groups show similar mean lines per code cell, suggesting no significant difference.

Number of Markdown Cells: Significant variation observed, potentially due to different teaching approaches across years.

Number of Attributes: CS students consistently have more attributes, with a large confidence interval in 2023-2024 due to potential notebook processing errors.

Number of Words in Markdown Cells: Significant difference in 2022-2023, indicating non-CS students might explain their processes in more detail.

Number of Lines of Code: Increasing trend for both groups, with non-CS students coding more verbosely.

Number of Code Cells: Similar to lines of code, an increasing trend, with non-CS students showing a significant difference in 2023-2024.

Cyclomatic Complexity: Overlapping confidence intervals, no significant conclusions drawn.

Mean Words in Markdown Cells: Irregularity in data, 2022-2023 as an outlier, suggesting similar metrics for both groups.

Number of Function Calls: Non-CS students significantly have more function calls, possibly influenced by previous class content.

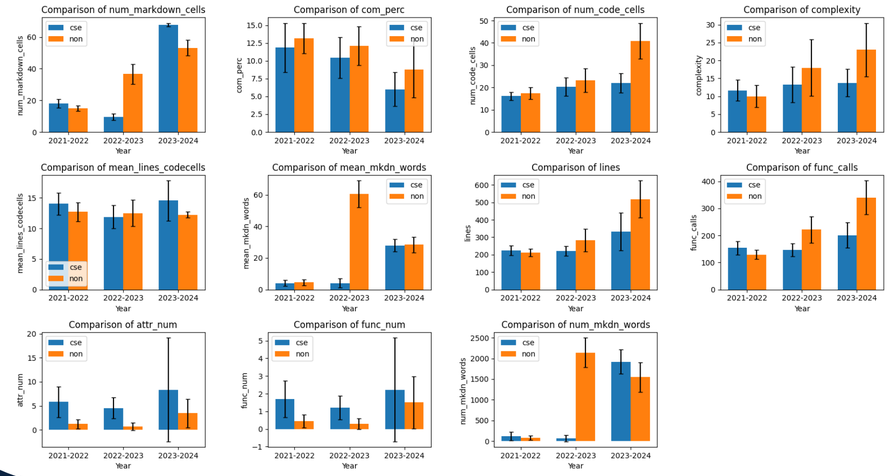
Number of Functions: Counterintuitively, CS students have more functions, but values are low for both groups.

There are 12 instances of a significant p-value (<0.05), with no metric having a significant p-value across all 3 years.

Relevant results include significant p-values for the number of functions and number of attributes for two consecutive years, indicating potential differences in algorithmic thinking.

The number of markdown cells also shows significant p-values, suggesting a tendency for students to explain their processes.

Significant p-value for the number of function calls indicates consistent usage by one group, likely involving library functions.



While most differences are not significant, noteworthy variations in attributes, functions, and markdown cells suggest potential distinctions in algorithmic thinking and explanation approaches between CS and non-CS students. Further investigation is needed to explore the reasons behind these trends.

4 Conclusion

Analyzing 60 ML assignments from the two student groups answers the research questions such that Computer Science students exhibit more modular thinking, creating variables for cleaner and reusable code, however, the values of other metrics showed no significant differences.

For the first and second sub-questions, the results show that there are rarely statistical differences in the metrics, meaning that one could not reliably distinguish the previous knowledge of a student based on their assignment. The answer to the third sub-question is no, because the comments and markdown cells for both groups do not have a significant difference.

To enhance the study's robustness, several considerations for future work are proposed:

- **Diverse Dataset:** Inclusion of submissions from different institutions and time frames, while ensuring assignment consistency, would provide a broader perspective on how students with varied backgrounds approach ML assignments.
- **Qualitative Analysis:** Introducing qualitative analysis, such as student interviews, could provide a deeper understanding of their coding approach and reasoning.
- **Controlled Experiments:** Implementing controlled experiments, where students solve problems under observation while explaining their thought process, would offer a more detailed insight into how different backgrounds influence problem-solving approaches.

5 Future work

Building upon the results of the study, there are several opportunities for further research.

- **A Comparative analysis of ML assignments:** An analysis of guided assignments, such as the ones in this paper, versus open-ended projects, which could highlight and improve the students computational thinking.
- **Exploration of prerequisite courses:** It would allow instructors to determine which knowledge from the courses applies to ML, and could be used to help tailor teaching methods and course content for students with different backgrounds.
- **Exploration of collaborative learning:** It would examine the effect of students of various backgrounds working on assignments together, investigating whether students with varying academic backgrounds could mutually positively affect their understanding of ML concepts and coding practices.

TU Delft