# What are the types of projects that Scratch users create?

## How do different types of features relate to the project type?

**Wojciech Marczuk**

**Supervisor(s): Fenia Aivaloglou, Sole Pera**

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Wojciech Marczuk
Final project course: CSE3000 Research Project
Thesis committee: Fenia Aivaloglou, Sole Pera, Jorge Martinez Castaneda

## Abstract

Scratch is a block-based programming language. It is designed to be simple and syntax error avoidant. This makes Scratch an accessible platform for cultivating coding skills. Many young learners are taught about different programming skills using various project types as examples. For instance, games are used as an engagement tool, and various games can motivate new learners to make their own. This influx of new published projects are manually classified into different types by using tags in the project descriptions. However, this manual classification only happens when the user adds the appropriate tags. This calls for an extension of that feature, which could help classify all projects published on the Scratch website. This has the goal of improving the browsing process, especially for new projects. To address this gap, in this initial iteration we leverage the fact that there are likely similarities, or even various project type defining features that would help improve the accuracy of classification through machine learning. Filling this gap also opens the possibility of automatic classification, depending on the accuracy of the results. Within this study, various machine learning models were tested with quantifiable project features as input data. The accuracy scores were compared to draw conclusions on how well various features extracted from Scratch projects performed for classification.


## 1 Introduction

The Scratch website has over 144 million projects published as of now[1]. While navigating the "explore" tab, the user can choose between different categories of projects, chosen by the creators of the projects themselves. This is a handy feature for browsing. However, it is possible that a project could be incorrectly categorised, or that the creator simply might not know which category fits their project best. We would also like to know about the preferences of different types of creators. Therefore, it may be useful to research and analyse how different projects among various categories differ on a technical level. It may also provide insight and trends on the types of projects that young learners create.

Several works have explored the types of projects that scratch users make. One particular study made use of a comprehensive data set consisting of 127 programs created by children, pre-classified into five distinct project types: [6]

- Animation
- Game
- Interactive Art
- Music and Dance
- Video Sensing

[1]Scratch statistics https://scratch.mit.edu/statistics/

In that study, there was a need for project classification into these types, to draw conclusions for the types of projects made based on gender.

The process of project classification could possibly be improved, or potentially automated, with insights on what are the key elements for different types of projects. Therefore, for this study, various quantifiable features will be extracted from different scratch projects. A labelled dataset will be used, with categories similar to that of the above mentioned study.

Scratch projects differ in how they are constructed. The quantifiable features will be extracted from the code of the project. For example, these can be amount of variables, amount of loops, or other coding elements. Such numerical values allow for the usage of different machine learning models for project classification. Different machine learning models will return performance percentage values, which allow for a convenient result comparison process.

Since we will be working with multiple features, the machine learning models can be trained using different combinations of features. Some features are more relevant to a specific project type, therefore the performance of the machine learning models will differ depending on which features are chosen for training. It is therefore worthwhile to test out a range of different features. In this study we will look at training the models using all possible pairs of features, and finally training the models using all extracted features.

To summarise, the analytical focal point of this research lies in discovering the correlation between various quantifiable features, and the respective project types. With this work, the aim is to answer the following research questions:

RQ: How do different types of features relate to the project type?

Sub-question 1: Which pair of features is best to predict projects for each project type?

Sub-question 2: Which type of project is is most accurately identified with the extracted features?

These questions will guide the research, and the discussion and other conclusions will be made referring back to the research question.

## 2 Related Work

Several works have explored the types of projects that scratch learners create. A three-day course was conducted by A. Funke and K. Geldreich that analysed the learning outcomes of an introductory programming course for primary school children [6]. During the course, 127 different scratch projects were created. The paper looks into the specific differences between the various projects created, and performs an analysis about their relation to gender. It was found that the types of projects created are indeed related to the gender of the project author. The project types that the projects were classified into, encapsulate a wide variety of project categories that are possible to create in scratch. Therefore, these project types were taken as inspiration for the choice of project types in this report.

This research was further extended by I. Graßl et al. with

an attempt to use an LDA model for project classification [7]. Such a model can be advantageous for this study, as it provides the probabilities for topic associations. Data labelling would also be less time consuming. However, as stated in their research, it is necessary to decide on the number of topics, or themes, in advance. This proves to be too large of a challenge with labelling from a very large data set choosing projects at random. Therefore, for this project, manual labelling will be done.

K. Amanullah and T. Bell address issues in programming habits of scratch users [2]. Their work gives valuable insight on specific features to analyse. A significant percentage of scripts contain dead code, and it is possible that the amount of dead code could relate to the project type. Various patterns were also found, relating to the frequency of blocks and structures used. It turns out that less than 3% of the projects analysed used lists. However, loops such as "repeat", or "forever" had frequencies of around 20% and 33% respectively. Features that appear more often should be prioritised in this study, since they should appear more commonly in the various projects chosen here for analysis.

T. McCabe's cyclomatic complexity measures the number of linearly-independent paths through a program module [3][5]. This could be a useful metric since there may be a large enough logical distinction within the different project types. Even though cyclomatic complexity has been critiqued in the past as being outperformed by other metrics [9], or measuring the same property as lines of code [8], it does give an insight on the complexity of the program, for which differences are likely depending on the project type.

## 3 Methodology

For accessibility and reproducibility of the methodology process, the dataset referred throughout the process, and machine learning training code used is publicly accessible as a github source[2]

### 3.1 Dataset

There is a lack of publicly available datasets that could be of use for the purpose of this study. Consequently, 107 randomly selected projects from a scratch dataset of scraped projects[3] [1] were used to create the dataset used for this research. This was an arbitrary number, chosen with the intention of having at least 100 projects to improve the accuracy of the future results as much as possible. Manual labelling was performed to classify the different projects into different categories. In figure 1, the project distribution is displayed. Since a manual labelling approach was decided on, it is important to have clear definitions of the categories. These definitions were rigorously followed when labelling each project to minimise human error:

- Game - A game is a project that a user can interact with, that has an end-goal. This end-goal can be something that can be achieved or controlled with inputs.

[2]Labelled dataset and code used for this project https://github.com/Nnomii/scratch-project-classification

[3]Scraped Scratch project dataset by Felienne Hermans and Fenia Aivaloglou https://github.com/TUDelftScratchLab/ScratchDataset

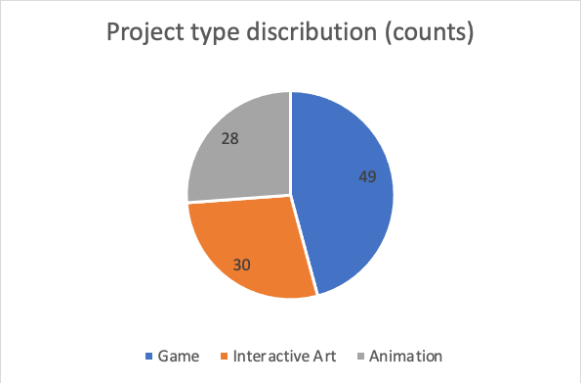


Figure 1: The distribution of projects by category in the dataset

- Animation - An animation is a project that, unlike a game, doesn't have user interaction for reaching an end goal. An animation showcases a series of user-independent frames that, when played together, showcase movement or other changes. Can also be defined as a short movie.
- Interactive Art - Interactive art projects allows for user interaction, while still being purposed to visually appeal. The user interaction could change how the art behaves, but unlike a game, there is no clear end purpose or progression. These projects are typically abstract.

### 3.2 Feature selection

All scratch projects within the dataset have already been scraped. The data of all the scratch projects can be downloaded, and is stored as a CSV file with rows corresponding to block information. From each row, it is possible to extract information about the type of block used and its parameter, if it exists. This allowed for the usage of SQL queries to extract relevant data.

Following research about the programming habits of scratch users [2], the high frequencies and variance in the amount of "repeat" and "forever" loops hint that they may be a useful metric to analyse. On the other hand, metrics that returned consistently low frequencies such as lists, are probably redundant for analysis, since for most projects the scores will be too similar, and thus not useful for the model's classification.

Cyclomatic complexity may also vary, to a high enough extent, depending on the type of project. For instance, it is expected that games are logically more complex than projects of different categories.

Therefore, with the help of the related work mentioned in section 2, these are the features that were extracted for machine learning:

- Cyclomatic Complexity
- Loop count
- Variable count
- "Say" block percentage

The CSV file of the scraped data contains all block (code) information for each project. To illustrate an example for feature extraction, this is the pseudocode process to calculate cyclomatic complexity:

1. Construct an SQL query that counts all occurrences of 'doIf' and 'doIfElse' blocks
2. Add 1 to the result
3. Group the counts by project ID

With this method, a new dataframe was created that was then later merged with project labels for each project ID.

### 3.3 Project type identification

Since each feature returns a quantifiable value, applying the data to machine learning algorithms becomes convenient. All machine learning models were trained with a 75% training data to 25% test data split. The following machine learning models were used, since they are popular choices for classification:

- Support Vector Machine (SVM)
- Decision Tree (DT)
- Random Forest (RF)

An analysis was made comparing the accuracy score of each machine learning model with the data. Since this is a multiclass classification problem, a OneVsRest (OvR) classifier approach was used [4]. For all projects of a certain type, every pair of features was fed into the three machine learning models, to see which pair performs best. The data from this will help answer sub-question 1.

To answer sub-question 2, the same experiment was run, but this time using all features as a four dimensional array.

## 4 Results

Some machine learning algorithms do a better job at classification than others, depending on many factors. However to answer the research question and sub-questions, it is imperative to compare how well the individual features predict the type of project, and to analyse the scores for all pairs. This will answer sub-question 1. The results from the machine learning models when all features were used, should be on average better performing. These results will be used to answer sub-question 2.

### 4.1 Decision boundary visualisation

The scatter graph in Figure 2 shows all 107 projects plotted for visualisation of the input data. Some projects have very similar values, resulting in overlapping points. The blue data points represent projects with the "game" label, and the red data points are all other projects. Points used as test data are semi-opaque.

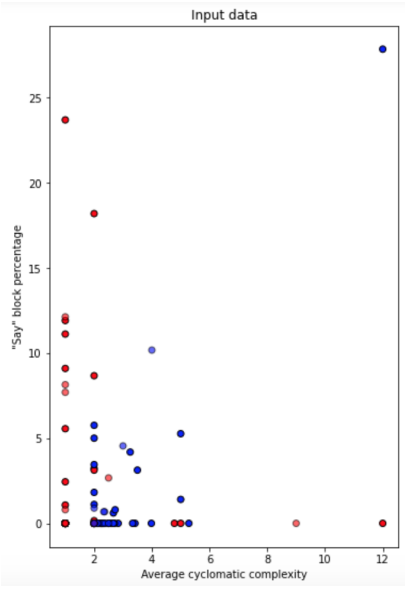


Figure 2: Scatter graph of the "say" block percentage values compared with average cyclomatic complexity.

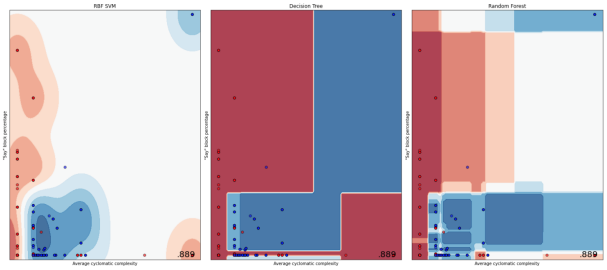


Figure 3: Machine learning model performances for the input data from figure 2.

Figure 3 shows the data from Figure 2, run through the three machine learning models. This visualisation is included to help better understand the decision boundaries of the machine learning models. The scores in the bottom right represent the accuracy score for the test data, and are of interest to this study.

### 4.2 Feature pair classification

The following tables represent the performance for every machine learning model for identifying projects using all possible feature pairs. Since an OvR classifier approach was used, the tables include the values for every machine learning model.

| GAME | Avg. CC | | | Loops | | | Variables | | | Say % | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF |
| Avg. CC | x | | | 0,889 | 0,815 | 0,852 | 0,889 | 0,778 | 0,815 | 0,889 | 0,889 | 0,889 |
| Loops | 0,889 | 0,815 | 0,852 | x | | | 0,667 | 0,704 | 0,741 | 0,556 | 0,481 | 0,519 |
| Variables | 0,889 | 0,778 | 0,815 | 0,667 | 0,704 | 0,741 | x | | | 0,667 | 0,741 | 0,741 |
| Say % | 0,889 | 0,889 | 0,889 | 0,556 | 0,481 | 0,519 | 0,667 | 0,741 | 0,741 | x | | |

Figure 4: Performance for each machine learning model for identifying games.

| ANIMATION | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. CC | | | Loops | | | Variables | | | Say % | | |
| | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF |
| Avg. CC | x | | | 0,815 | 0,778 | 0,741 | 0,815 | 0,741 | 0,741 | 0,926 | 0,852 | 0,889 |
| Loops | 0,815 | 0,778 | 0,741 | x | | | 0,704 | 0,704 | 0,704 | 0,741 | 0,63 | 0,667 |
| Variables | 0,815 | 0,741 | 0,741 | 0,704 | 0,704 | 0,704 | x | | | 0,741 | 0,63 | 0,704 |
| Say % | 0,926 | 0,852 | 0,889 | 0,741 | 0,63 | 0,667 | 0,741 | 0,63 | 0,704 | x | | |

Figure 5: Performance for each machine learning model for identifying animation projects.

| INTERACTIVE ART | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. CC | | | Loops | | | Variables | | | Say % | | |
| | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF | SVM | DT | RF |
| Avg. CC | x | | | 0,778 | 0,852 | 0,815 | 0,778 | 0,513 | 0,741 | 0,778 | 0,704 | 0,741 |
| Loops | 0,778 | 0,852 | 0,815 | x | | | 0,778 | 0,704 | 0,852 | 0,778 | 0,815 | 0,741 |
| Variables | 0,778 | 0,513 | 0,741 | 0,778 | 0,704 | 0,852 | x | | | 0,778 | 0,852 | 0,741 |
| Say % | 0,778 | 0,704 | 0,741 | 0,778 | 0,815 | 0,741 | 0,778 | 0,852 | 0,741 | x | | |

Figure 6: Performance for each machine learning model for identifying interactive art projects.

The following graphs represent the average performance for identifying projects across the three machine learning models.

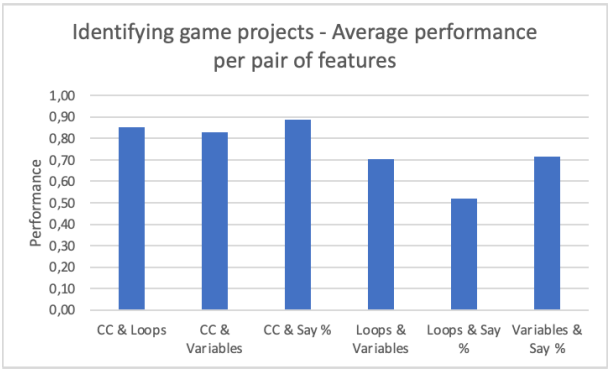


Figure 7: Average performance for the machine learning models for identifying games.

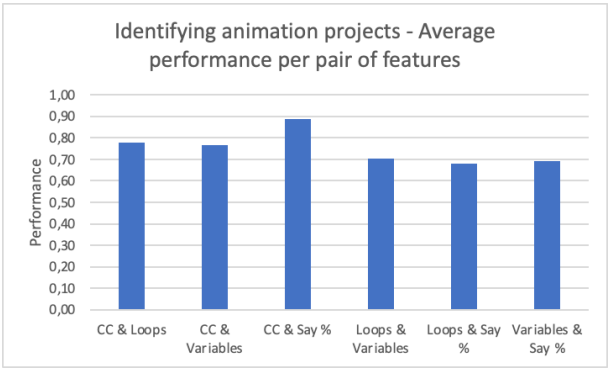


Figure 8: Average performance for the machine learning models for identifying animation projects.

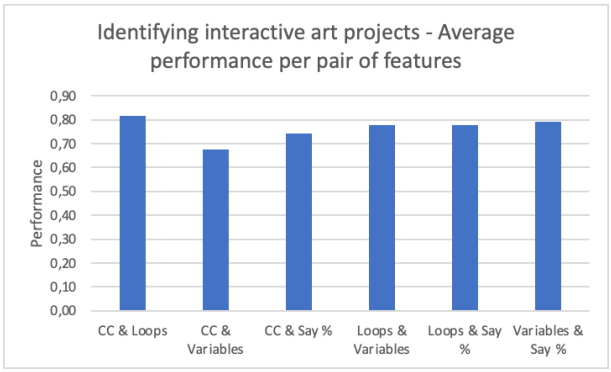


Figure 9: Average performance for the machine learning models for identifying interactive art projects.

### 4.3 All-feature classification

Finally, the three machine learning models were tested with all data from the extracted features. The results are represented in figure 10.

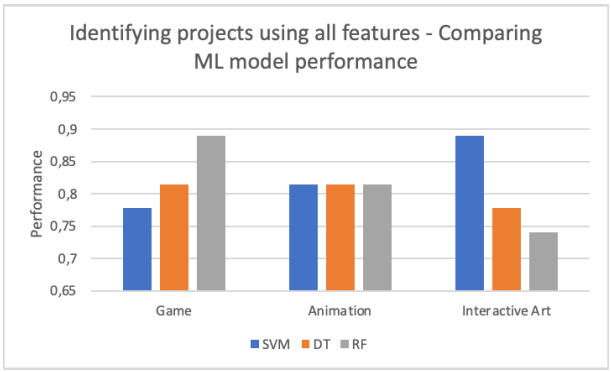


Figure 10: Performance of different machine learning models using all extracted features to classify projects.

## 5 Responsible Research

All research done in this project adheres to the five responsible research principles. Principle number one is honesty, and this concerns refraining from data fabrication and avoiding unfounded claims. The methodology in this report is reproducible, as the dataset and all code used can be publicly accessed and the exploration process is clearly described in the methodology section.

Principle number two is scrupulousness. This project follows this research principle. This is thanks to a combination of research into papers with similar themes, along with following machine learning conventions. As a result, the methods used are scientific and scholarly.

Principle number three is transparency. The data collection process is explained step by step. As stated before, all code and data can be accessed (reference again). Since manual dataset labelling was performed, the grounds of how each label was applied to each scratch project is described, with clear label definitions. All steps in the research process are verifiable.

Principle number four is independence. In particular, the methodology, and all sections after are not influenced by non-scholarly articles, nor do they contain non factual information such as any commercial or political nature.

Principle number five is responsibility. The data collection does not cross any ethical matters. All projects used are taken from the scratch public database, to which users share publicly. No other parts of the research and investigation process impact people, animals or the environment and are therefore considered responsible.

# 6 Discussion

## 6.1 Feature pair classification

For classifying game projects, cyclomatic complexity and the percentage of "say" blocks was the feature pair that had the best performance of 89%, see figure 4 and figure 7. Coincidentally, this top score is visualised by the previous example in figures 2 and 3. From both of these figures, the scatter graph and the decision boundaries for the different machine learning models, show why this pair of features performed well. The vertical line of red points at where the average cyclomatic complexity equals one, shows that there were no games in that region from our dataset. There is also a large group of blue points having a lower amount of "say" blocks, but a higher cyclomatic complexity than most other non-game projects. All of this was recognised by each machine learning model. The trend of games having a larger cyclomatic complexity than other projects is logically evident, as games typically have many steps or paths to take during runtime.

For classifying animation projects, one pair of features stood out and this is once again cyclomatic complexity and the percentage of "say" blocks with a top performance of 89%, see figures 5 and 8. Cyclomatic complexity proved to be a great feature at recognising games, therefore leading to easier recognition of other projects, such as animation projects in this case. During manual labelling we noticed that many animations consisted of the character sprites talking with each other through "say" blocks, and this was also part of the motivation for this feature choice.

For classifying interactive art projects, the top performing pair of features this time was cyclomatic complexity and the number of loops with a performance of 82%, see figures 6 and 9. This is slightly worse than for classification of the other two types of projects. Additionally, observing the bar chart in figure 9, there is a lower variance in the classification performance for the different machine learning models compared to the classification of the other two types of projects. This shows that the features selected are less significant metrics for the interactive art category. Since the top performance is also slightly worse, a reasonable conclusion is that there may exist more significant metrics for this project type, or that interactive art projects are just harder to classify due to their broad definition of what is considered "art". It is likely that this time, the number of loops performed better as a part of the feature pair, due to interactive art projects almost always making use of multiple forever loops that repeat actions until the user interacts, stopping the process or starting another loop.

For all three project types, it is interesting to note that cyclomatic complexity was part of the top scoring pair of features. This is most likely due to how certain project types rely on certain conditions of this feature to be satisfied. For instance, it is difficult to imagine a game with a cyclomatic complexity of 1, meaning that there is only one independent path to take through the programmed logic. Another interesting takeaway is the visibly low classification performance for game projects using the pair of number of loops and the percentage of "say" blocks. This low score of 52% is exceptionally low and almost similar to random guessing for classification, since 47% of projects in the input data were games, see figure 1. This shows that the combination of say blocks along with loops are probably poor choices of project defining features for games.

## 6.2 All-feature classification

Figure 10 shows the classification performance while using all four features. For games, random forest performed best. For animation projects, all models performed equally. For interactive art, support vector machine performed best. Since the results vary for each project type, it is not possible to decide on a best performing machine learning model for classification. They all perform similarly enough, and to analyse why some models could be better for the classification of specific project types would be too specific and out of the scope of this research.

## 6.3 Limitations

Considering a 25% test data to 75% training data split that was used for this project, this resulted in just under 27 test data points from the dataset of 107 projects. This is arguably the minimal data sample size to return meaningful results. The reliability of the results could be further improved by using a much larger labelled dataset of Scratch projects.

Even though the best performance of certain machine learning models scored an high accuracy of $\approx$90%, one could argue that this score is still not good enough, if this automatic classification were to be used by Scratch on their main website for new projects. While it is interesting to see feature similarities, incorrect classification of projects should be avoided. Therefore, a model like this can instead be used to suggest categories for to the user, for newly created projects.

# 7 Conclusion

## 7.1 Summary

The research question for this study concerned how different types of features relate to the project type. It followed from the results that the best pair of features for classification slightly differed depending on what type of project the classification was done on. For games, the best pair of features was cyclomatic complexity, and the "say" block percentage. For animation projects, the best pair of featues was also cyclomatic complexity, and the "say" block percentage. For interactive art, the best scoring pair of features was cyclomatic complexity, and the number of loops. Cyclomatic complexity was a part of every top scoring feature pair, making it a good metric for classification.

There is no clear answer to what type of project is easiest to classify from the features used in this study. As illustrated in figure 10, the performance is too similar when classifying projects using all extracted features.

### 7.2 Future work

Since there was no clear answer to the question of what project is easiest to identify, it is possible that a better choice of features could be chosen. More research can be done into the similarities of programming projects that fall into these categories, not only made in Scratch. This can give a better idea into what features are most relevant to extract for classification. It is also likely that a larger combination of different features can be used, expanding on the four used in this study. These improvements can lead to better accuracy scores for classification. Automated project classification could be justified, if the accuracy scored for classification scores were large enough.

## References


[1] Efthimia Aivaloglou, Felienne Hermans, Jesus Moreno-Leon, and Gregorio Robles. A dataset of scratch programs: Scraped, shaped and scored. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 511–514, 2017.

[2] Kashif Amanullah and Tim Bell. Analysing students' scratch programs and addressing issues using elementary patterns. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2018.

[3] IBM documentation. Cyclomatic complexity. https://www.ibm.com/docs/en/raa/6.1?topic=metrics-cyclomatic-complexity, March 2021.

[4] Scikit-learn documentation. Multiclass and multioutput algorithms. https://scikit-learn.org/stable/modules/multiclass.html, 2023.

[5] Christof Ebert, James Cain, Giuliano Antoniol, Steve Counsell, and Phillip Laplante. Cyclomatic complexity. *IEEE software*, 33(6):27–29, 2016.

[6] Alexandra Funke and Katharina Geldreich. Gender differences in scratch programs of primary school children. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 57–64, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Isabella Graßl, Katharina Geldreich, and Gordon Fraser. Data-driven analysis of gender differences and similarities in scratch programs. In *The 16th Workshop in Primary and Secondary Computing Education*, pages 1–10, 2021.

[8] Jay Graylin, Joanne E Hale, Randy K Smith, Hale David, Nicholas A Kraft, WARD Charles, et al. Cyclomatic complexity and lines of code: empirical evidence of a stable linear relationship. *Journal of Software Engineering and Applications*, 2(03):137, 2009.

[9] Martin Shepperd. A critique of cyclomatic complexity as a software metric. *Software Engineering Journal*, 3(2):30–36, 1988.