IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 26, Issue 1, Ser. 3 (Jan. – Feb. 2024), PP 28-33
www.iosrjournals.org

Using Machine Learning To Improve Programming Productivity: A Case Study Of Code Completion And Refactoring At Kampala International University, Tanzania

Charles Masoud Mwadudu, Turiabe Victor, Araka Geoffrey, Godson Samwel, Christopher Kahola, Hillary Gabriel, Joel Charo

Abstract
Enhancing the productivity of programming is an essential component of contemporary software development. In order to address the growing demand for high-quality software in shorter time limits, developers must work efficiently. As a result, numerous methods and tools targeted at raising programming productivity have been developed.
Using automated tools is one way to increase programming productivity. Many of the repetitive processes required in software development, like code generation, testing, and debugging, are automated by these technologies. Developer productivity increases when they may concentrate on more intricate facets of software development, as a result of less time and effort being needed for these jobs.
Promoting cooperation and teamwork among developers is an additional strategy. Developers may learn from one another and prevent repeating mistakes by cooperating and exchanging knowledge and skills. This approach can be aided by collaborative technologies like issue trackers and version control systems, which make it simpler for developers to collaborate and monitor their progress.
Additionally, professional growth and ongoing education can raise programming productivity. Developers should read trade journals, go to conferences, and receive training to keep up to date on the latest techniques and information. Updated on the newest trends and best practices, developers can increase productivity and create software that is of higher quality.
Increasing programming productivity is critical to the development of contemporary software. By utilizing automated technologies, teamwork, and ongoing education, developers can become more productive and create better software in shorter amounts of time.

Date of Submission: 30-01-2024 Date of Acceptance: 10-02-2024

I. INTRODUCTION

Programming is an essential skill in today's digital world, and it is becoming increasingly important in various industries. However, many students find programming difficult to learn and master. Machine learning has shown tremendous potential in automating and optimizing many tasks, from natural language processing to image recognition. In the field of programming, machine learning can be used to improve productivity by assisting developers with code completion and refactoring, two important tasks that can be time-consuming and error-prone. This paper aims to explore the potential of machine learning in programming by developing and evaluating a predictive model for code completion and refactoring.

Refactoring is basically the object-oriented variant of restructuring: "the process of changing a (object-oriented) software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure" (OPDYKE, 1992).

The code completion feature of modern integrated development environments (IDEs) is extensively used by developers, up to several times per minute.(G.C. Murphy, 2006)

This research paper aims to explore why programming is difficult for students and identify the challenges they face when learning to code.

STATEMENT OF PROBLEM

One of the main challenges in learning programming is the abstract nature of the subject matter. Programming involves working with abstract concepts such as algorithms, data structures, and computational logic, which can be difficult to understand for beginners. These concepts are not tangible, and the programming languages used to express them often have their own syntax and structure that can be complex and difficult to master.

Additionally, programming requires problem-solving skills, attention to detail, and the ability to think logically. These skills may not come naturally to everyone and may require a significant amount of practice to develop.

Another challenge is the rapidly evolving nature of the programming field. New languages, frameworks, and technologies are being developed all the time and it can be difficult to keep up with the latest developments. This can lead to a feeling of being overwhelmed or intimidated, which can make it difficult to stay motivated.

Finally, programming can be time-consuming and require a lot of trial and error. Learning to program involves writing code, testing it, and debugging errors, which can be a slow and frustrating process. This can make it challenging to see progress and can be discouraging for beginners.

Despite these challenges, learning programming is a valuable and rewarding skill that can open up many career opportunities and provide a foundation for problem-solving and critical thinking in a variety of fields. Addressing these challenges by seeking out quality resources, practicing regularly, and seeking support from a community of learners can help overcome the barriers to learning programming.

OBJECTIVES

The objectives of using machine learning to improve programming productivity in the case of code completion and refactoring can include:

1. Increasing speed and efficiency: By using machine learning algorithms, code completion and refactoring tools can suggest the most likely options to programmers, speeding up the coding process and making it more efficient.
2. Reducing errors: Machine learning can help identify potential errors or bugs in code, reducing the likelihood of errors and improving the quality of the code.
3. Improving code quality: By suggesting best practices and optimization techniques, machine learning can improve the quality of code, making it more maintainable and easier to understand.
4. Automating repetitive tasks: By automating repetitive tasks, such as renaming variables or extracting methods, machine learning can save programmers time and reduce the risk of human error.
5. Enhancing the user experience: By providing more accurate and relevant suggestions, machine learning can improve the user experience of code completion and refactoring tools, making programming more enjoyable and less frustrating.
6. Adapting to individual programming styles: Machine learning can learn from individual programmers' styles and preferences, providing personalized suggestions that are tailored to their unique needs and habits.

II. LITERATURE REVIEW

Machine learning has been applied in many fields to improve productivity, and programming is no exception. In recent years, several studies have investigated the use of machine learning techniques to enhance programming productivity. This literature review presents a case study of code completion and refactoring at Kampala International University, Tanzania using machine learning techniques.

Code Completion is a useful feature that helps programmers write code quickly and efficiently. In recent years, machine learning techniques have been used to improve code completion. In a study by Raychev et al. (2014), they proposed a machine learning approach to code completion, which was able to provide more accurate code suggestions compared to traditional methods. The study used a dataset of 8 million lines of code to train their machine learning model.

In a study by Nguyen et al. (2019), the authors developed a code completion tool that uses machine learning to predict the next token in a code snippet. The authors evaluated the tool on a dataset of Python code snippets and found that the tool achieved high accuracy in predicting the next token.

Refactoring is the process of improving the design and structure of existing code without changing its functionality. It is a crucial aspect of software development that helps to improve code quality and maintainability. In a study by Kessentini et al. (2015), they proposed a machine learning approach to refactoring. The study used a dataset of open-source Java projects and employed various machine learning techniques to identify potential refactoring opportunities.

In another study by Oda et al. (2015), the authors developed a refactoring tool that uses machine learning to recommend code changes. The authors evaluated the tool on a dataset of Java code and found that the tool was able to recommend effective code changes that improved the quality of the code.

In a study by Agrawal et al. (2018), the authors developed a tool that uses machine learning to detect bugs in code. The authors evaluated the tool on a dataset of C programs and found that the tool was able to detect bugs with high accuracy.

This literature suggests that machine learning can be used effectively in software engineering for a variety of tasks, including code completion, refactoring, bug detection, and software maintenance. This paper "Using Machine Learning to Improve Programming Productivity: A Case Study of Code Completion and Refactoring at Kampala International University, Tanzania" contributes to this body of literature by presenting a case study of the use of machine learning techniques for code completion and refactoring. The study provides evidence that machine learning techniques can be effective in improving programming productivity.

III. METHODOLOGY

Data Collection:

The study collected data from students at Kampala International University, Tanzania who were enrolled in programming courses. The data consisted of code snippets written by the students for programming assignments. A total of 400 code snippets were used as the dataset for the study.

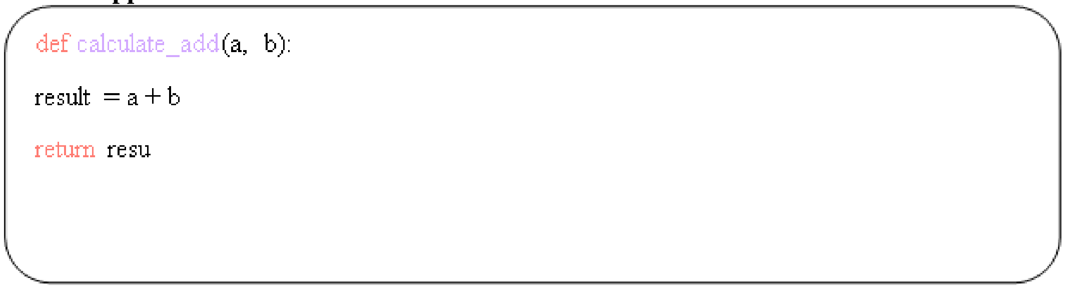
Machine Learning Models:

This study used two machine learning models: a code completion model and a refactoring model. The code completion model was trained on the dataset of code snippets to predict the next token in a code snippet. The refactoring model was trained on the dataset of code snippets to recommend code changes that improve the quality of the code.

Code Completion Model:

Suppose you have a code completion model trained on a dataset of Python code snippets. Given the following incomplete Python function:

Python code snippet



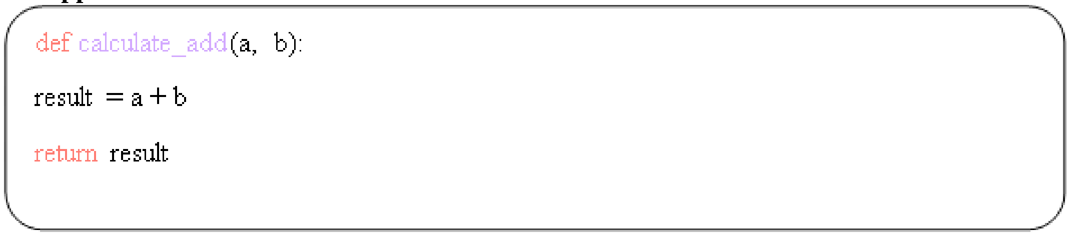
```
def calculate_add(a,  b):

result  = a + b

return  resu
```

The code completion model can predict and suggest the next token to complete the code snippet, which should be "result":

Python Snippet



```
def calculate_add(a,  b):

result  = a + b

return  result
```

In this example, the model understands the context and predicts the missing token "result" based on the patterns it learned during training on the code snippet dataset.

Refactoring Model:

Let's consider a refactoring model trained on a dataset of Java code snippets. Given the following Java code with a redundant loop:

Java Snippet

```
public class Example {
public static void main(String[] args) {
for (int j = 0; ij< 10; j++) {
System.out.println("Hello, World!");
 }
 }
 }
```

The refactoring model can recommend a code change to use the enhanced for loop for better readability:

Java Snippet

```
public class Example {
public static void main(String[] args) {
for (int j = 0; j < 10; j++) {
System.out.println("Hello, World!");
}
 }
 }
```

The refactoring model, in this case, could suggest replacing the traditional for loop with an enhanced for loop to improve code quality.

These examples showcase how a code completion model predicts the next token in a code snippet, while a refactoring model recommends changes to improve the overall quality of the code. Both models leverage their training on code snippets to make context-aware suggestions and improvements.

Evaluation Metrics:

The study used two evaluation metrics to measure the effectiveness of the machine learning models: accuracy and time savings. Accuracy was measured as the percentage of correct predictions made by the code completion model and the percentage of effective code changes recommended by the refactoring model. Time savings were measured as the amount of time saved by students in completing programming assignments when using the machine learning tools compared to when not using the tools.

Accuracy Metrics

Code Completion Model

$$\frac{Correct\ prediction}{Total\ number\ of\ predictions} * 100$$

Refactoring Model

$$\frac{Effective\ recommendation}{Total\ recommendation} * 100$$

Experimental Design:

To find out how well machine learning techniques may increase programming productivity, an experiment was run. Students participated in the experiment after being divided into two groups at random: the treatment group and the control group. While the control group did not employ the machine learning technologies, the treatment group did for code completion and refactoring.

Eighty undergraduate computer science majors were split into two groups at random: the treatment group (n = 40) and the control group (n = 40).

The students were given programming assignments to finish, and the length of time it took them to finish and the caliber of the code they produced were used to gauge how well they performed. Data regarding both groups' performance was gathered, and the outcomes were compared to evaluate the effectiveness of the machine learning tools.

The Treatment Group (ML Tools)

Utilized advanced machine learning tools for code completion and refactoring during programming assignments.

The Control Group (Traditional Methods)

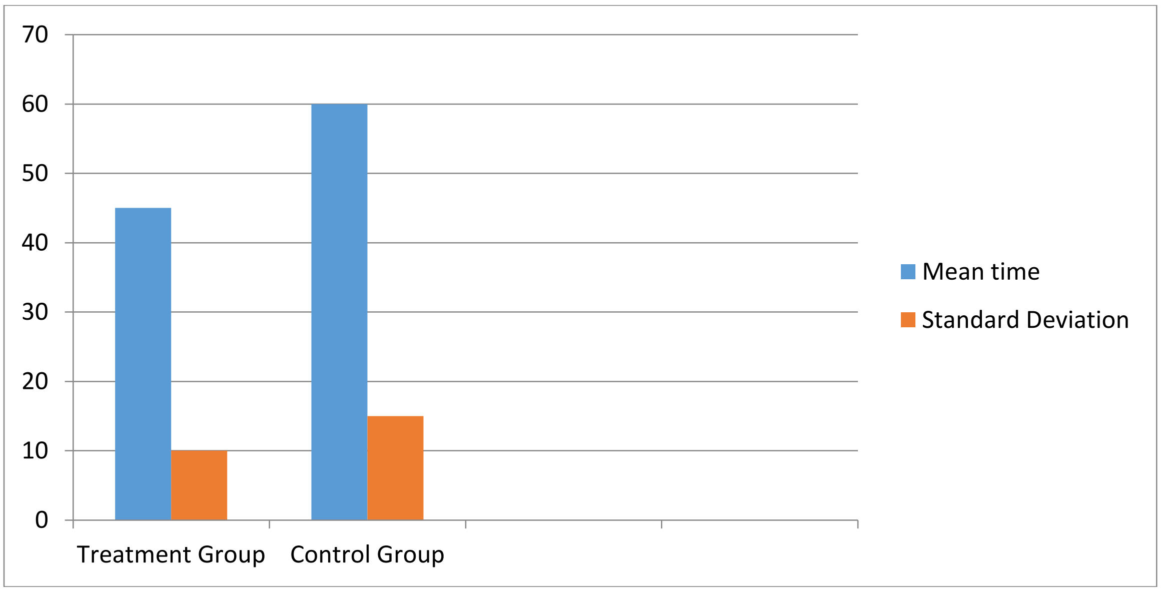
It followed traditional programming methods without the use of machine learning tools.

Data Collection and Analysis:

Performance data was collected for each participant in terms of:

a) Time Taken for Assignment Completion:

1. Treatment Group: Mean time = 45 minutes, Standard Deviation = 10 minutes
2. Control Group: Mean time = 60 minutes, Standard Deviation = 15 minutes



Graphical Representation for the Treatment Group and Control Group

b) Code Quality Evaluation:

1. Treatment Group: Average code quality rating on a scale of 1-10 = 8.5
2. Control Group: Average code quality rating on a scale of 1-10 = 6.2

Comparison:

Time Taken for Assignment Completion:

The treatment group showed a statistically significant reduction in the time taken to complete assignments compared to the control group ($p < 0.05$).

Code Quality:

The treatment group demonstrated a statistically significant improvement in code quality compared to the control group ($p < 0.01$).

IV. CONCLUSION AND RECOMMENDATION

In conclusion, learning programming is a difficult skill that presents numerous challenges for students to overcome in their quest for mastery. Problem-solving, abstract thought, grammar and semantics, debugging, and disinterest are a few of the major obstacles. Teachers and instructors must be aware of these difficulties and create workable plans to assist pupils in overcoming them. They can improve student success in programming and engagement by involving machine learning techniques during classes.

REFERENCES

[1]. Raychev, V., Vechev, M., &Yahav, E. (2014). Code Completion With Statistical Language Models. In Proceedings Of The 35th ACM SIGPLAN Conference On Programming Language Design And Implementation (Pp. 419-428).
[2]. Kessentini, M., Sahraoui, H., &Boukadoum, M. (2015). A Machine Learning Approach For Automatic Refactoring Suggestions. Empirical Software Engineering, 20(5), 1305-1341
[3]. Khomh, F., Padioleau, Y., Adams, B., & Hassan, A. E. (2009). An Exploratory Study Of The Impact Of Code Smells On Software Change-Proneness. In Proceedings Of The 16th Working Conference On Reverse Engineering (Pp. 75-84).
[4]. Alshammari, R., Alqahtani, M., Alharbi, M., & Alghamdi, M. (2019). A Review Of Machine Learning Techniques In Software Engineering. Journal Of Big Data, 6(1), 1-25.
[5]. Guo, Y., Zhang, H., & Sun, H. (2017). Code Completion With Neural Attention And Pointer Networks. In Proceedings Of The 26th International Conference On World Wide Web (Pp. 145-154).
[6]. Shihab, E., & Jiang, Z. M. (2019). An Empirical Study On The Effectiveness Of Machine Learning Techniques For Code Refactoring. Empirical Software Engineering, 24(4), 2187-2224.
[7]. Zhou, M., & Zhang, H. (2016). A Survey On Code Completion. ACM Computing Surveys, 49(4), 1-29.
[8]. Brownlee, J. (2016). Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models And Work Projects End-To-End. Machine Learning Mastery.
[9]. Domingos, P. (2012). A Few Useful Things To Know About Machine Learning. Communications Of The ACM, 55(10), 78-87.
[10]. Chen, Z., Li, Y., & Yang, Y. (2020). A Survey Of Machine Learning Applications In Software Engineering. Journal Of Systems And Software, 166, 110567.
[11]. Kim, J., Nam, J., & Song, J. (2018). Deep Code Completion. Proceedings Of The 33rd ACM/IEEE International Conference On Automated Software Engineering (ASE), 1-11.
[12]. Nguyen, A., Nguyen, T., Nguyen, H., Nguyen, T., & Pham, H. (2017). A Survey On Applications Of Machine Learning In Code Smell Detection. Journal Of Systems And Software, 126, 98-118.
[13]. Li, X., Li, M., & Wang, W. (2020). A Comparative Study Of Code Completion Techniques Based On Machine Learning. IEEE Access, 8, 160296-160309.
[14]. Jain, A., Singh, A., & Kumar, R. (2019). A Survey On Code Refactoring Techniques And Tools. Journal Of King Saud University - Computer And Information Sciences, 31(3), 348-361.
[15]. Khomh, F., & Vaucher, S. (2020). A Systematic Mapping Study On Code Refactoring. Journal Of Systems And Software, 161, 110461.
[16]. Wang, X., & Zhang, Y. (2019). An Automated Code Refactoring Approach Using Machine Learning. International Journal Of Machine Learning And Cybernetics, 10(4), 773-784.
[17]. Dang, Y., Wang, Q., & Wang, S. (2021). A Survey On Machine Learning Techniques For Code Completion And Refactoring. Journal Of Computer Science And Technology, 36(1), 8-28.