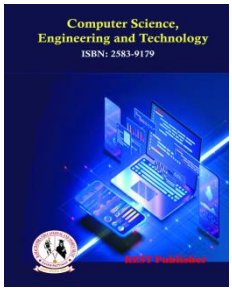
Computer Science, Engineering and Technology
Vol: 3(3), September 2025
REST Publisher; ISSN: 2583-9179 (Online)
Website: https://restpublisher.com/journals/cset/
DOI: https://doi.org/10.46632/cset/3/3/12



Measuring AI-Driven Developer Productivity in Agile Software Development: Random Forest Regression Analysis of Performance Metrics and Tool Integration

Ranjan Kumar Mishra, Rajan kumar
Netaji Subhas University Jamshedpur, India.
Corresponding Author Email: ranjanlnmi4u@gmail.com

Abstract: *The rapid integration of generative artificial intelligence (AI) tools in software development has created an urgent need to understand their impact on developer productivity and project outcomes. By examining AI-driven performance metrics for agile software development, this research addresses the current gap in empirical evidence on the effectiveness of generative AI in handling complex, real-world software projects. Traditional productivity metrics such as lines of code (LOC), feature completion rates, and defect density are inadequate for measuring productivity in AI-assisted environments. The study proposes new metrics including AI-assisted code generation, accuracy in defect detection, reduced developer effort, improved code quality, and reduced cognitive load to better assess the transformative potential of AI tools such as Git Hub Co-pilot and Open AI Codex. The research explores how these tools are reshaping traditional development workflows by automating routine tasks, enabling early bug detection, and improving project management processes. By analysing data from code repositories, issue trackers, and collaboration tools, the study provides evidence-based insights into developer productivity patterns. The findings show that AI-powered development tools can significantly reduce manual effort, accelerate software delivery, and improve overall quality, while allowing developers to focus on high-level design and strategic problem-solving, ultimately addressing the productivity paradox in modern software engineering.*

Keywords: *AI for development, Developer productivity, AI-assisted development, Software engineering metrics, Agile development, Code generation, AI-pair programming, Performance measurement, Software development tools and Productivity paradox.*

1. INTRODUCTION

The ongoing debate about using generative AI tools to increase productivity, improve efficiency, reduce errors, and accelerate software development has brought new attention to the study of productivity in this field. This research is driven by the current gap in empirical evidence regarding the effectiveness of generative AI in handling complex, real-world software projects [1]. As a result, new development approaches and methodologies are needed to address the high failure rates of IT projects. Generative AI-powered (gen AI) development tools offer the potential to improve both productivity and quality in software engineering. Gen AI generally refers to transformer-based language models that use artificial intelligence to understand and generate human language in text format. The so-called "productivity paradox" creates uncertainty for organizations, leaving them uncertain about how to effectively use gen AI. However, AI has the potential to automate routine tasks in software development and testing, analyse large datasets to find patterns, and systematically evaluate information within neural networks. These capabilities can speed up development cycles, reduce costs, and improve efficiency, while enhancing the creative potential of human developers [2]. By leveraging technologies such as machine learning, natural language processing, and predictive analytics, AI provides valuable insights into the factors that influence developer performance. By analysing data from code repositories, issue trackers, version control systems, and collaboration tools, AI can uncover patterns and trends that may go unnoticed by human observers. This helps teams better understand developer activities, pinpoint inefficiencies, and identify communication gaps. AI's ability to process and interpret this information in real time provides Agile teams with evidence-based insights that allow them to refine workflows and make informed changes, rather than relying on assumptions [3]. Traditional software metrics often emphasize output indicators such as lines of code (LOC), feature completion

rate, and defect density. However, measuring productivity in an AI environment requires additional metrics that reflect the benefits of automation, such as reduced developer effort, improved code quality, and reduced cognitive load. Metrics such as AI-assisted code generation, accuracy in defect detection, and time to complete tasks provide unique insights into the impact of AI on software development. These metrics are essential for assessing productivity in AI-supported environments. Another key objective is to propose scalable frameworks to effectively measure developer productivity in the AI era [4]. AI-powered tools are reshaping traditional development workflows by helping developers write code more efficiently, identify bugs early, and improve project management processes. This shift to AI-assisted development reduces manual effort, speeds up software delivery, and improves overall quality. AI brings significant benefits to software engineering, especially by automating routine tasks. Tools like Git Hub Co-pilot and Open AI Codex provide smart code suggestions that help reduce development time and reduce the likelihood of bugs. Additionally, AI-based debugging and testing systems improve software reliability by detecting issues early in the development process [5]. As the pressure for faster software delivery increases, traditional hand-coding methods often fail to meet business needs. Low-code platforms help address this challenge by simplifying the development process, although they still rely on developers to write and debug code for custom features. AI-powered code generation improves these platforms by automating the coding workload to a greater extent, so that developers can focus on high-level design and strategic problem-solving [6]. If not thoughtfully designed, AI-based performance metrics can lead to unexpected biases or misinterpretations, which can lead to inaccurate estimates or additional stress on developers. This study explores how AI-driven performance metrics can improve agile software development by providing objective, real-time insights into developer productivity. It explores the benefits, challenges, and best practices of using AI for performance measurement, while highlighting the need to balance automation with human intelligence and judgment [7]. In recent years, AI technologies have been increasingly integrated into many areas of software development, including code generation and project management. This introduction lays the groundwork for exploring how AI is reshaping software development practices. By exploring the synergy between human creativity and machine intelligence, this research seeks to reveal the impact of AI on software engineers and development teams, providing insights into its transformative potential within the industry [8]. In software engineering, a significant portion of research and development focuses on AI-assisted tools designed to improve developers' ability to produce high-quality code. With the growing acceptance of AI for code generation, numerous studies have sought to assess how effectively these tools can support developers in real-world situations [9]. AI-pair programming is a collaborative approach to software development in which a human developer works with an AI assistant on the same codebase. The concept draws inspiration from traditional pair programming, where two human developers collaborate by taking on the role of a driver who writes the code, while the other acts as a navigator who reviews it and provides guidance [10].

2. MATERIALS AND METHODS

Hours Coding Per Day: Hours of coding per day refers to the average number of hours a developer actively spends writing, reviewing, or debugging code during a typical workday. It is a productivity metric used to assess developer engagement and workload, help teams understand time allocation, identify potential burnout risks, and optimize work distribution for improved software development efficiency and team performance.

Number Tools Used: Number of tools used refers to the total number of software tools, platforms, or applications that a developer uses during the software development process. This includes code editors, version control systems, debugging tools, testing frameworks, project management software, and collaboration platforms. Tracking this metric helps assess tool diversity, workflow complexity, and potential performance or integration challenges, providing insights into the developer's work environment and the overall development ecosystem.

LOC per Day: Lines per day (LOC) is a metric that measures the average number of lines of code a developer writes in a workday. It is used to measure coding activity and productivity. However, it should be interpreted with caution, as a higher LOC count does not necessarily indicate better performance, as code quality, complexity, and maintainability are equally important factors in assessing software development performance.

Tasks Completed Per Week: Tasks Completed per Week is a productivity metric that tracks the number of assigned development tasks, such as coding, debugging, testing, or documentation, that are successfully completed by a developer or team within a week. It provides insight into workflow efficiency, individual or team output, and progress toward project goals. While useful for tracking performance and planning, this metric should also be considered along with task complexity and quality to avoid inflated estimates of productivity.

Instructions for machine learning

Random Forest Regression: Random forest regression is an ensemble learning method designed to predict continuous values. It builds multiple decision trees using randomly selected subsets of both the data and features, then combines their outputs by averaging the results. This approach reduces the risk of over fitting and improves the generalizability of the model. The randomness between trees introduces heterogeneity, which leads to more

reliable and accurate predictions than a single tree. Random forest regression is well suited to capturing complex, nonlinear relationships and managing high-dimensional datasets. It is resilient to noise and outliers and also provides valuable insights into feature importance, revealing the most influential variables. Its reliability, flexibility, and high accuracy make it a popular choice for predictive modelling in fields such as finance, healthcare, and engineering.

3. RESULTS AND DISCUSSIONS

"AI Developer Productivity" provides insights into various productivity metrics for AI developers, measured across four key parameters: hours coded per day, number of tools used, lines of code per day (LOC), and tasks completed per week. Each row represents data from a different developer or work session. Hours coded per day ranged from about 3.4 to 7.9 hours, with most developers coding between 5 and 7 hours. The number of tools used ranged from 2 to 5, indicating that developers were using multiple tools to improve productivity. LOC per day a common measure of developer output varied widely from 158 to 364, reflecting differences in coding speed, project complexity, and the nature of tools used. Tasks completed per week ranged from 6 to 25, a wide range that again underscores individual variation in work speed and task complexity. Interestingly, higher LOC is often associated with more hours of coding and a higher number of tools. For example, developers who code more than 7 hours and use 4–5 tools have more than 300 LOC and complete more tasks. This indicates a possible positive correlation between tool usage, time investment, and productivity.

TABLE 1. Descriptive Statistics

	Hours Coding Per Day	Number Tools Used	LOC per Day	Tasks Completed Per Week
count	100	100	100	100
mean	5.898	3.64	244.52	17.36
std	0.910764	1.150494	38.24574	3.483061
min	3.4	2	158	6
25%	5.4	3	222.75	15
50%	5.9	4	248	17
75%	6.425	5	265.5	19.25
max	7.9	5	364	25

Table 1 provides descriptive statistics for the 100 participants, highlighting daily coding time, tools used, lines of code written (LOC), and weekly task completion. On average, participants coded for 5.9 hours daily using approximately 3.6 tools, generated approximately 245 LOC per day, and completed 17.4 tasks per week. Coding hours ranged from 3.4 to 7.9, while tools used varied from 2 to 5. LOC output ranged from 158 to 364, and weekly tasks ranged from 6 to 25, indicating moderate variability in the measurements.

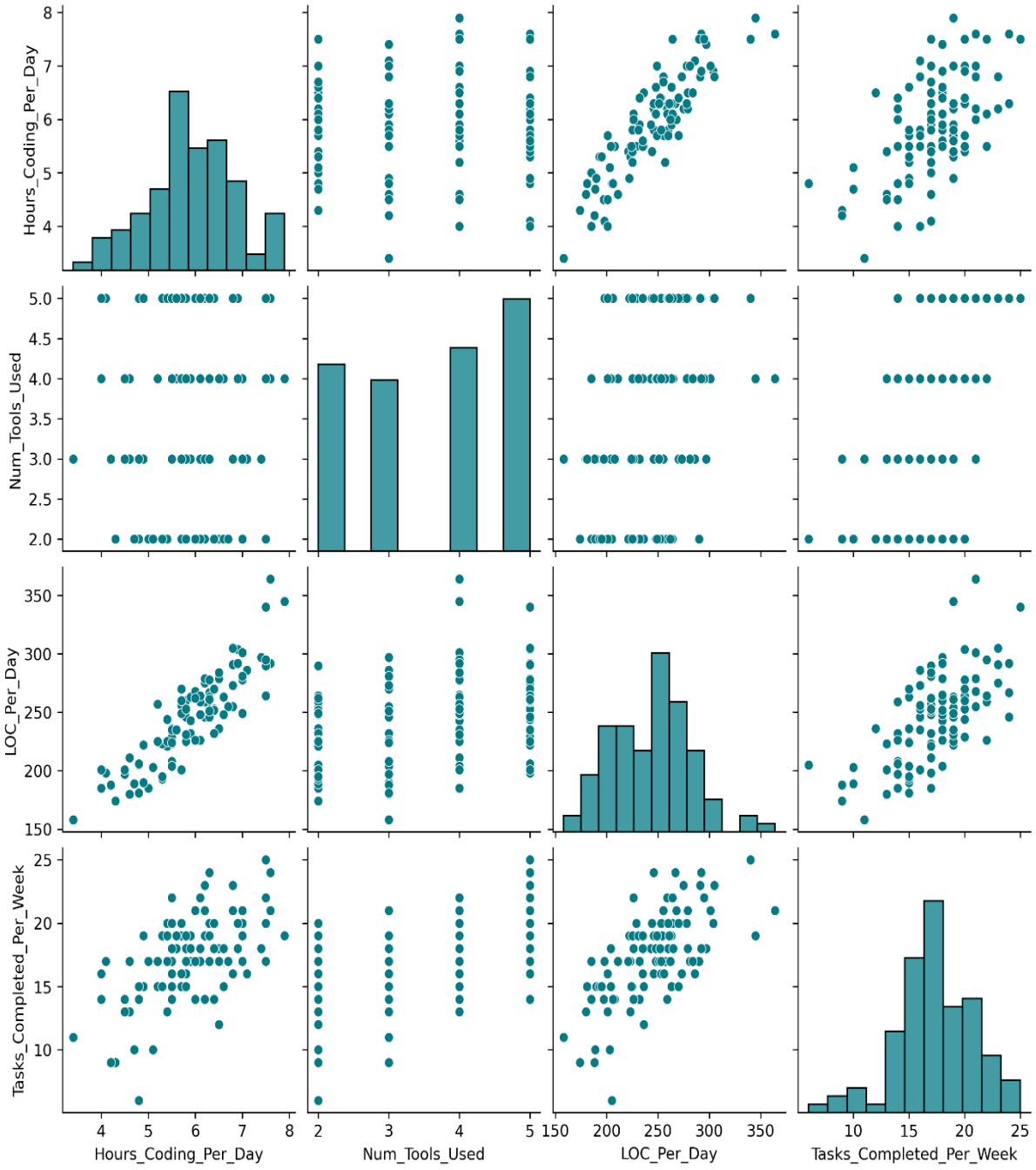


FIGURE 1. Scatter plot of the various AI Developer Productivity Effect of Process Parameters

Figure 1 is a scatter plot matrix showing the relationships between various AI developer productivity metrics, including hours of coding per day, number of tools used, code queues per day, and tasks completed per week. In particular, strong positive correlations are visible between coding time, LOC, and weekly task completion, indicating consistent productivity patterns.

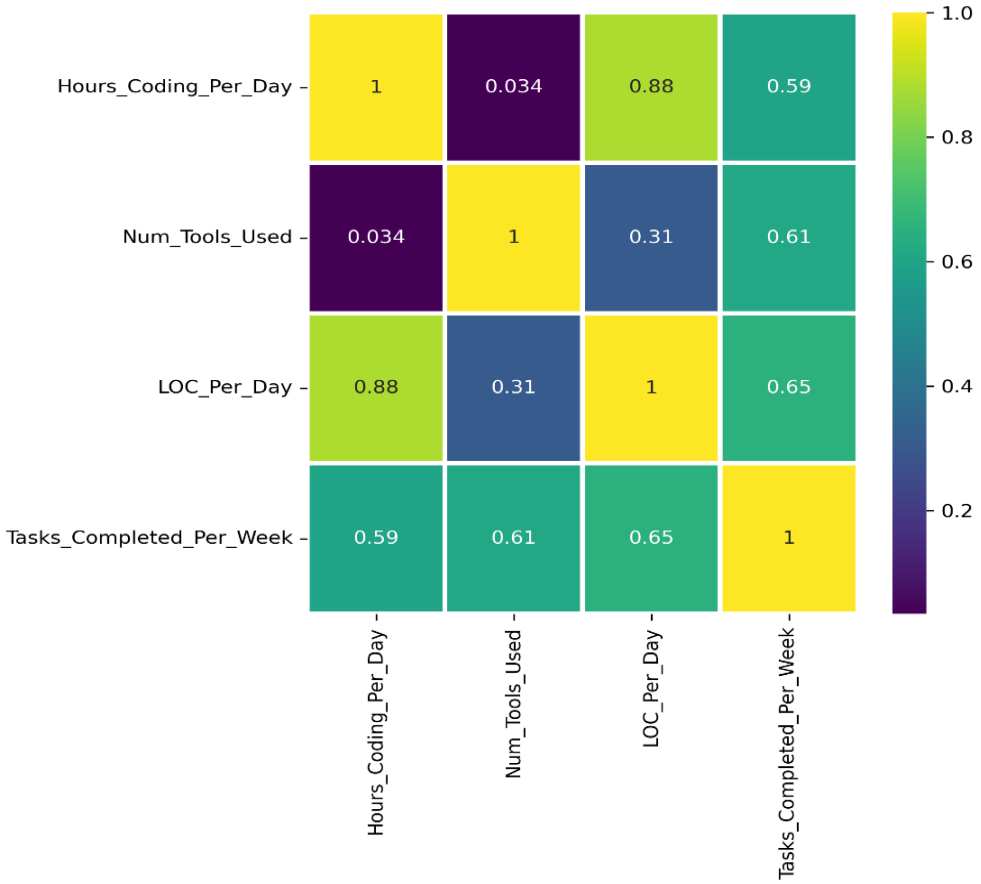


FIGURE 2. Heat map of the connection between process variables and outcomes

Figure 2 is a heat map illustrating the relationship between AI developer productivity variables. Hours of coding per day and LOC per day exhibit a strong positive relationship (0.88), while tasks completed per week shows moderate correlations with all variables. In particular, numerical tools used have a moderate correlation with tasks completed (0.61), indicating that diversified tool use may help productivity.

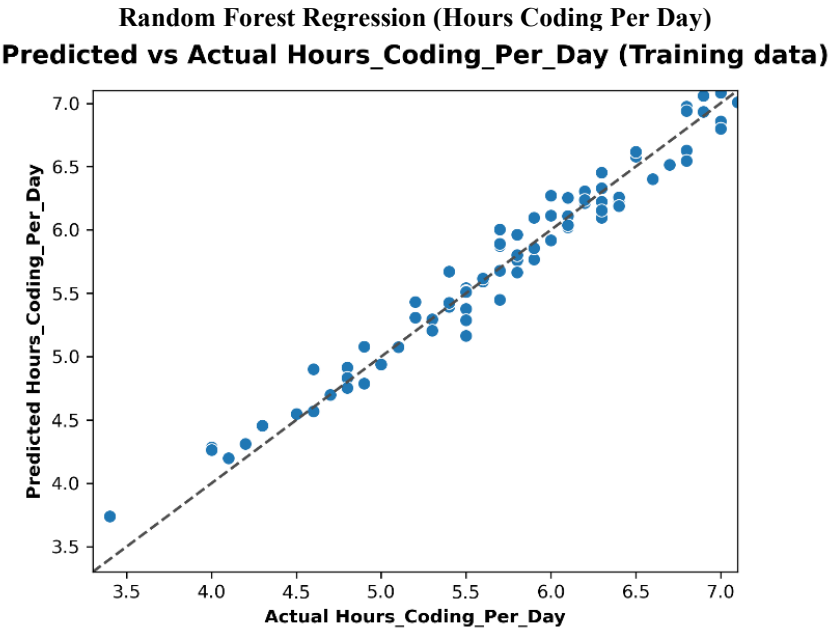


FIGURE 3. Random Forest Regression on Hours Coding per Day: training data

Figure 3 Scatter plot of predicted and actual hourly index per day using random forest regression on the training data. Close alignment of data points on the diagonal reference line indicates a high level of predictive accuracy, indicating that the model effectively captures underlying patterns in developer productivity.

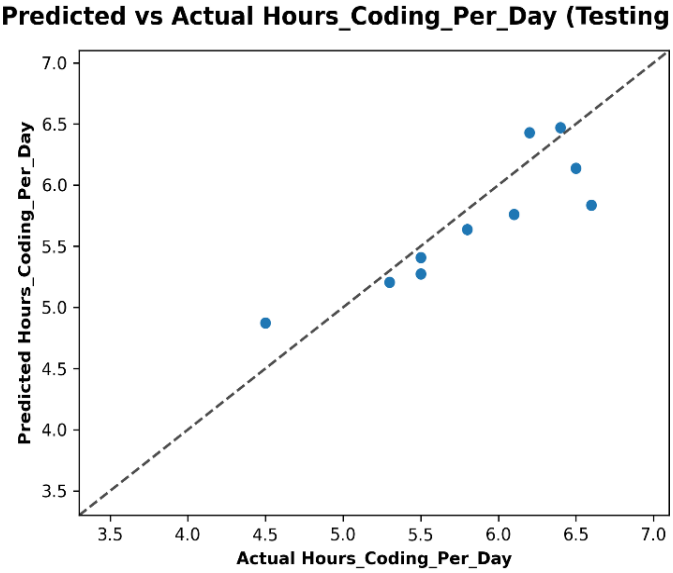


FIGURE 4. Random Forest Regression on Hours Coding per Day: testing data

Figure 4 Scatterplot comparing predicted and actual hourly index per day using random forest regression on the test data. While the predictions are generally consistent with the actual values, small deviations from the diagonal indicate some variability. Overall, the model maintains reasonable accuracy, validating its performance on unobserved data.

TABLE 2. Performance Metrics of Random Forest Regression on Hours Coding per Day (Training Data and Testing Data)

Property	Data	Symbol	Model	R2	EVS	MSE	RMSE	MAE	Max Error	MSLE	Med AE
Hours Coding per Day	Train	RFR	Random Forest Regression	0.97306	0.97307	0.02342	0.15305	0.12413	0.33900	0.00055	0.11700
	Test	RFR	Random Forest Regression	0.70845	0.75675	0.11207	0.33477	0.27125	0.76350	0.00237	0.22750

Table 2 summarizes the performance of the random forest regression (RFR) model in predicting daily coding times. On the training data, this model achieved high accuracy with an R^2 of 0.973, a mean square error (MSE) of 0.02342, and a root mean square error (RMSE) of 0.153. In contrast, performance on the test data showed reduced accuracy, with an R^2 of 0.708, an MSE of 0.112, and an RMSE of 0.335, suggesting some over fitting but acceptable generalization.

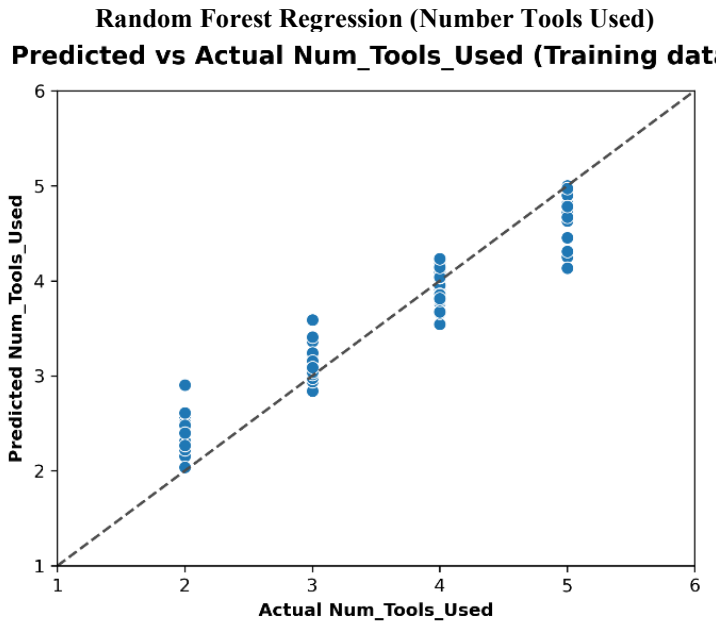


FIGURE 5. Random Forest Regression on Number Tools Used: training data

Figure 5 is a scatterplot showing the difference between the predicted numbers of tools used using Random Forest Regression on the training data and the actual number. Most of the data points closely follow the diagonal, indicating strong predictive accuracy. The unique nature of the variable is clear, and the model effectively captures the usage pattern of tools among developers.

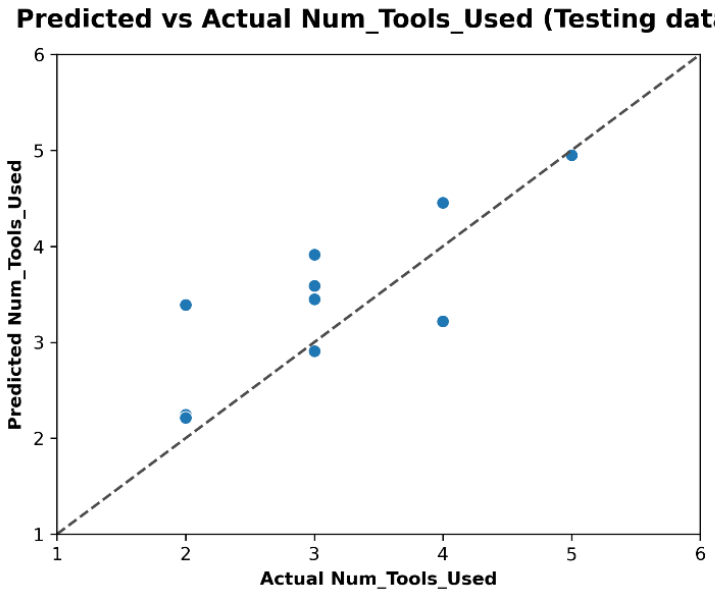


FIGURE 6. Random Forest Regression on Number Tools Used: testing data

Figure 6 Scatter plot of the predicted and actual number of tools used from random forest regression on the test data. Although the predictions are generally aligned, significant deviations from the diagonal indicate reduced accuracy compared to the training set. The model captures overall trends, but shows small discrepancies with discrete categorical values.

TABLE 3. Performance Metrics of Random Forest Regression on Number Tools Used (Training Data and Testing Data)

Property	Data	Symbol	Model	R2	EVS	MSE	RMSE	MAE	MaxError	MSLE	MedAE
Number Tools Used	Train	RFR	Random Forest Regression	0.92954	0.92990	0.09309	0.30511	0.22617	0.90500	0.00580	0.18500
	Test	RFR	Random Forest Regression	0.52223	0.64757	0.42522	0.65209	0.51800	1.39000	0.02656	0.45250

Table 3 shows the performance of the random forest regression model in predicting the number of tools used. On the training data, the model performed well, with an R^2 of 0.93, a low MSE of 0.093, and an RMSE of 0.305. However, on the test data, the performance decreased significantly, with an R^2 of 0.522 and an RMSE of 0.652. Increased error metrics, including high MAE and maximum error, indicate over fitting and reduced predictive reliability in the missing data.

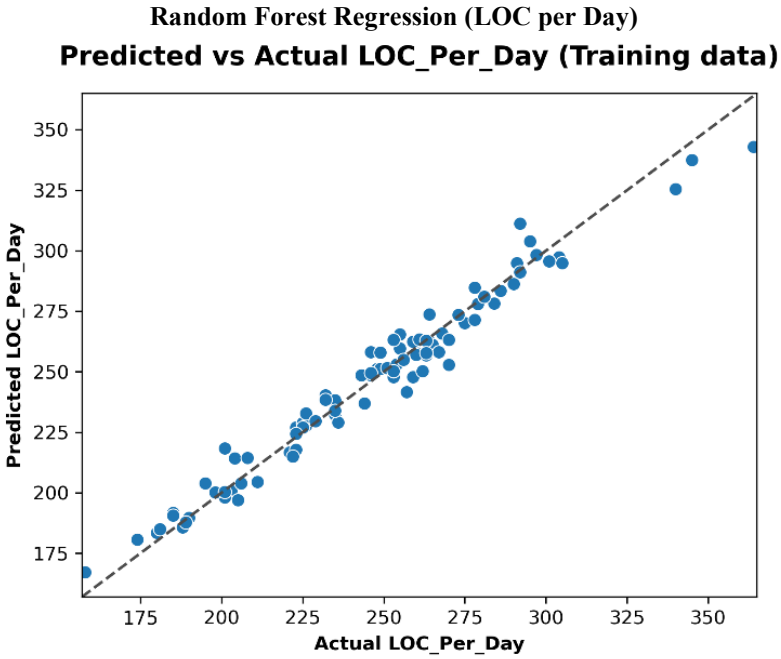


FIGURE 7. Random Forest Regression on LOC per Day: training data

Figure 7 Scatterplot of predicted and actual lines of code (LOC) per day using random forest regression on the training data. The points closely follow the diagonal reference line, demonstrating excellent model accuracy. This strong alignment indicates that the model effectively captures the coding productivity patterns of AI developers during training.

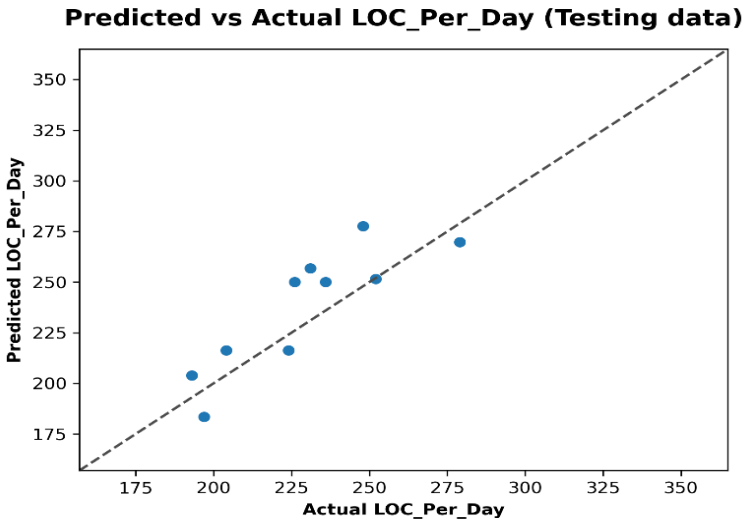


FIGURE 8. Random Forest Regression on LOC per Day: testing data

Figure 8 is a scatterplot comparing predicted and actual line of code (LOC) per day using random forest regression on the test data. The plot exhibits moderate alignment with the diagonal, indicating good predictive performance. Although some predictions deviate, the model captures trends in code productivity in the data that are not typically observed.

TABLE 4. Performance Metrics of Random Forest Regression on LOC per Day (Training Data and Testing Data)

Property	Data	Symbol	Model	R2	EVS	MSE	RMSE	MAE	MaxError	MSLE	MedAE
LOC per Day	Train	RFR	Random Forest Regression	0.96664	0.96666	50.29920	7.09219	5.53546	21.05500	0.00079	4.49750
	Test	RFR	Random Forest Regression	0.54589	0.66102	291.63168	17.07723	14.76857	29.56900	0.00509	12.91750

Table 4 presents the performance metrics of the Random Forest Regression (RFR) model for predicting lines of code per day (LOC). On the training data, the model performs exceptionally well with a high R^2 (0.96664), low mean square error (MSE), and root mean square error (RMSE), indicating accurate predictions. However, on the test data, the performance drops significantly, with an R^2 of 0.54589 and high error values in metrics such as MSE, RMSE, and mean absolute error (MAE), indicating possible over fitting.

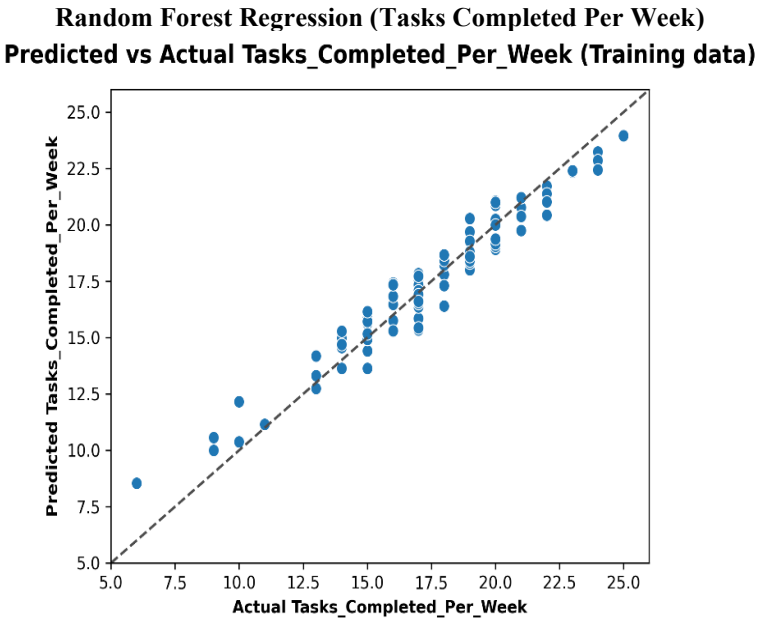


FIGURE 9. Random Forest Regression on Tasks Completed per Week: training data

Figure 9 illustrates the performance of the random forest regression model in predicting the number of tasks completed per week based on training data. The model analyses various features to capture patterns and trends, demonstrating its ability to accurately estimate weekly task completion rates during training.

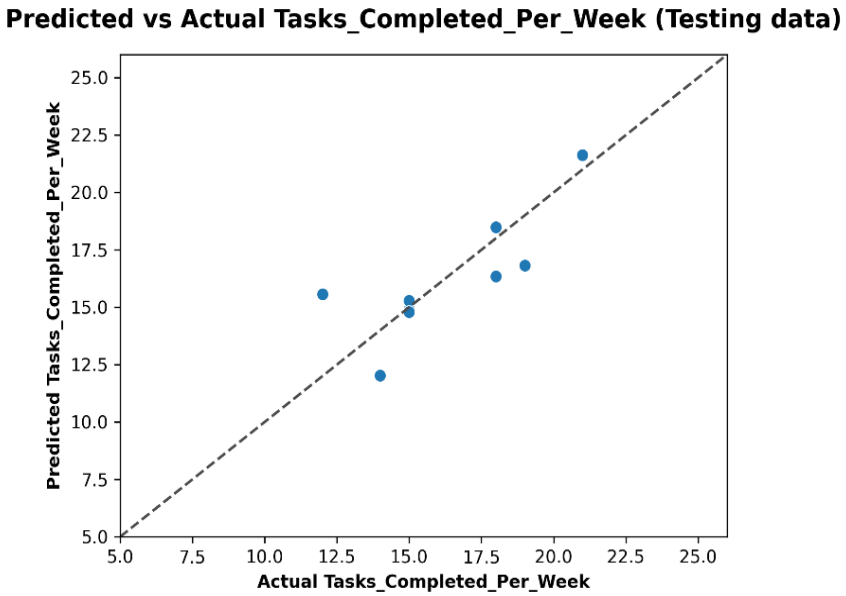


FIGURE 10. Random Forest Regression on Tasks Completed per Week: testing data

Figure 10 shows the results of the random forest regression model used to test the data for tasks completed per week. It assesses the predictive accuracy of the model on unobserved data, highlighting its generalization ability and effectiveness in estimating weekly task completion outside the training set.

TABLE 5. Performance Metrics of Random Forest Regression on Tasks Completed per Week (Training Data and Testing Data)

Property	Data	Symbol	Model	R2	EVS	MSE	RMSE	MAE	MaxError	MSLE	MedAE
Tasks Completed Per Week	Train	RFR	Random Forest Regression	0.93776	0.93776	0.77887	0.88254	0.73917	2.55000	0.00356	0.68250
	Test	RFR	Random Forest Regression	0.58724	0.59844	2.74489	1.65677	1.26800	3.57000	0.01100	1.12000

Table 5 summarizes the performance of the random forest regression (RFR) model in predicting the number of tasks completed per week. The model shows strong performance on the training data with a high R^2 value of 0.93776 and low error measures such as MSE (0.77887) and RMSE (0.88254). However, the test data results reveal reduced accuracy, with R^2 decreasing to 0.58724 and high error values including RMSE at 1.65677 and MAE at 1.26800, indicating some over fitting and reduced generalization ability.

4. CONCLUSION

Based on a comprehensive analysis of AI-driven performance metrics in agile software development, this research demonstrates that AI tools for development are significantly changing developer productivity patterns and measurement approaches. By introducing new metrics that capture the nuanced impact of AI assistance beyond traditional measures such as lines of code and defect density, this study successfully addresses an identified gap in empirical evidence regarding AI performance on real-world software projects. Findings reveal strong correlations between key productivity indicators, with hours coded per day and LOC per day showing a significant 0.88 correlation coefficient, while tool diversity is moderately associated with task completion (0.61). These relationships suggest that AI-driven development environments produce synergistic effects, where increased tool usage and coding time investment amplify overall productivity outcomes. Random forest regression models achieved high accuracy on all metrics (R^2 values ranging from 0.93 to 0.97) on training data, although generalization to test data showed the expected performance degradation, indicating the difficulty in predicting human-AI joint productivity. Research confirms that AI tools such as Git Hub Co-pilot and Open AI Codex successfully automate routine tasks, reduce cognitive load, and help developers focus on strategic problem-solving rather than machine coding activities. However, the study also highlights the critical need for balanced implementation, as AI-based metrics require careful design to avoid biases and misinterpretations that can create additional developer stress. This investigation establishes a foundation for scalable frameworks that measure developer productivity in the AI era, moving beyond the traditional productivity paradox toward evidence-based insights. The proposed metrics of AI-assisted code generation accuracy, reduced developer effort, and improved code quality provide organizations with operational tools to improve AI integration. Future research should focus on examining long-term productivity trends and developing standardized benchmarks for AI-assisted development environments to ensure sustainable and beneficial adoption of production AI technologies in software engineering practices.

REFERENCES

[1]. Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. "The impact of ai on developer productivity: Evidence from github copilot." arXiv preprint arXiv: 2302.06590 (2023).

[2]. Dohmke, Thomas, Marco Iansiti, and Greg Richards. "Sea change in software development: Economic and productivity analysis of the ai-powered developer lifecycle." arXiv preprint arXiv: 2306.15033 (2023).

[3]. Weisz, Justin D., Shraddha Vijay Kumar, Michael Muller, Karen-Ellen Browne, Arielle Goldberg, Katrin Ellice Heintze, and Shagun Bajpai. "Examining the use and impact of an ai code assistant on developer productivity and experience in the enterprise." In Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, pp. 1-13. 2025.

[4]. Moroz, Ekaterina A., Vladimir O. Grizkevich, and Igor M. Novozhilov. "The potential of artificial intelligence as a method of software developer's productivity improvement." In 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), pp. 386-390. IEEE, 2022.

[5]. Coutinho, Mariana, Lorena Marques, Anderson Santos, Marcio Dahia, Cesar França, and Ronnie de Souza Santos. "The role of generative ai in software development productivity: A pilot case study." In Proceedings of the 1st ACM International Conference on AI-Powered Software, pp. 131-138. 2024.

[6]. Li, Mahei Manhai, Ernestine Dickhaut, Olivia Bruhin, Hendrik Wache, and Pauline Weritz. "More than just efficiency: Impact of generative AI on Developer productivity." (2024).

[7]. Asad, Shahzaib, and Zhao Feng. "AI in Agile Workflows: Measuring Developer Productivity with Data-Driven Insights." (2023).
[8]. SOLANKE, ADEDAMOLA ABIODUN. "Software Development in the Age of AI: Quantifiable Frameworks for Measuring Enhanced Developer Productivity."
[9]. Sager, Javid, and Abdul Wahab. "Enhancing Software Development with AI: Measuring Productivity and Automation Impact." (2024).
[10]. Hassan, Mohammad Asif. "Impact of adopting AI tools by software developers towards productivity and sustainability." (2024).
[11]. Thompson, Jace. "Enhancing Developer Productivity with AI-Driven Code Generation in Low-Code Platforms." (2024).
[12]. Razzaq, Abdul, Jim Buckley, Qin Lai, Tingting Yu, and Goetz Botterweck. "A Systematic Literature Review on the Influence of Enhanced Developer Experience on Developers' Productivity: Factors, Practices, and Recommendations." ACM Computing Surveys 57, no. 1 (2024): 1-46.
[13]. Viswanadhapalli, Vamsi. "AI-Augmented Software Development: Enhancing Code Quality and Developer Productivity Using Large Language Models." (2024).
[14]. Szikszó, Zoltán Péter. "Artificial Intelligence for Software Developers." (2024).
[15]. Haque, Ebtesam Al, Chris Brown, Thomas D. LaToza, and Brittany Johnson. "Towards Decoding Developer Cognition in the Age of AI Assistants." arXiv preprint arXiv: 2501.02684 (2025).
[16]. Batte, Benjamin. "The evolving landscape of code review: Leveraging artificial intelligence for enhanced software quality and developer productivity." Available at SSRN (2025).
[17]. Jordan Nelson, Victoria Daniel. "Personalized Developer Experience through AI." (2025).
[18]. Cui, Zheyuan Kevin, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. "The effects of generative ai on high skilled work: Evidence from three field experiments with software developers." Available at SSRN 4945566 (2024).
[19]. Edelman, Benjamin G., Donald Ngwe, and Sida Peng. "Measuring the impact of AI on information worker productivity." Available at SSRN 4648686 (2023).
[20]. Asproni, Giovanni. "Hans Dockter on Developer Productivity." IEEE Software 42, no. 01 (2025): 129-132.

