Understanding the Effects of GenAI as No-Code Alternative for Teaching Machine Learning Workflows

Martin Strobel
Singapore Polytechnic
martin.r.strobel@gmail.com

Abstract

Teaching machine learning (ML) workflows to non-programmers remains a challenge in introductory AI courses. Traditionally, educators have turned to no-code tools such as KNIME to lower barriers. With the rise of generative AI (GenAI), students can now construct ML pipelines through natural language prompts, potentially offering a new "no-code" pathway. In a polytechnic-wide elective in Singapore, students were given the choice of using either KNIME or a GenAI chatbot for practical exercises and their semester project. Survey responses, informal interviews, and classroom observations revealed that both tools supported conceptual learning, but students' experiences diverged: KNIME provided predictability and structured guidance, while GenAI offered speed and flexibility yet posed setup challenges and required coding familiarity. Students valued having a choice, though this complicated teaching logistics. Our experience suggests that GenAI can complement—but not yet replace—traditional no-code platforms, and that the design of introductory activities is critical for adoption. We share lessons learned for educators considering GenAI as an alternative in workflow-based ML education.

Introduction

Artificial intelligence (AI) has moved rapidly from a specialized field into a technology shaping many aspects of society (Polak and Anshari 2024). As demand grows for AI literacy (Flynn 2024), educators face the challenge of making machine learning (ML) accessible to students from diverse backgrounds. A particular difficulty lies in teaching ML workflows to learners without programming experience: while coding offers flexibility, it presents a steep learning curve that can discourage non-specialists.

No-code and low-code tools such as KNIME (Berthold et al. 2009), Weka (Hall et al. 2009), and Orange (Demšar et al. 2013) have helped address this challenge by allowing students to construct ML pipelines through graphical interfaces. These tools lower barriers but also have drawbacks: students must learn a new interface, and classroom time may shift toward "click-by-click" procedures rather than conceptual understanding (Wang and Wang 2024).

The rise of generative AI (GenAI) introduces a new possibility: students can describe a workflow in natural language and obtain executable code, effectively treating GenAI as a no-code alternative. In this paper, we report on an exploratory study in a Singapore polytechnic elective where students chose between KNIME and GenAI for practical exercises and their final project. Our aim is to share observations, student feedback, and practical lessons on how GenAI can complement or challenge established no-code approaches in teaching ML workflows.

The remainder of this paper is organized as follows: we first situate our work in related literature on no-code AI and GenAI in education, then describe the course and study design. We present student survey results and qualitative feedback, followed by reflections on teaching with dual tools. We conclude with key lessons for educators and directions for future iterations.

Related Research

No-Code and Low-Code AI in Education

No-code and low-code solutions have attracted increasing attention in recent years, both in industry and education. They enable non-programmers with little or no coding experience to build functional software and machine learning applications (Bhattacharyya and Kumar 2023; Lethbridge 2021; Luo et al. 2021; Sahay et al. 2020; Sundberg and Holmström 2024; Yan 2021). Within enterprises, adopting low-code approaches has been shown to reduce development costs and time, while narrowing the gap between technical and business teams (Rokis and Kirikova 2022).

In educational contexts, no-code machine learning platforms such as KNIME, Orange, or Weka provide drag-and-drop interfaces that allow students to focus on concepts such as data preprocessing, model selection, and evaluation, without being blocked by programming challenges (Wang and Wang 2024; Sundberg and Holmström 2024). This has made them especially attractive in introductory AI courses and in outreach efforts targeted at students with diverse disciplinary backgrounds. By lowering technical barriers, these platforms have been argued to democratize access to machine learning and broaden participation across disciplines (How et al. 2021).

While no-code AI tools lower barriers for beginners, research has noted that they also introduce new challenges. On the practical side, such platforms can generate additional ad-

ministrative overhead for instructors and students, for example in terms of platform setup, account management, and organizing collaborative work (Sundberg and Holmström 2024). On the pedagogical side, tool-centric learning risks an overemphasis on mastering the mechanics of a particular software at the expense of developing broader skills such as critical thinking or collaborative problem-solving (Wang and Wang 2024). These observations highlight that while no-code AI can broaden access, their effective use in education requires careful instructional design to avoid both logistical and conceptual pitfalls.

GenAI in Education

Parallel to the development of no-code platforms, generative AI has recently emerged as a transformative technology for programming support. With the release and rapid global adoption of ChatGPT at the end of 2022, followed by many similar offerings, generative AI became widely accessible to students and educators. This accessibility has sparked both enthusiasm and concern, positioning GenAI in education as a young but rapidly growing field of study.

Discussions have centered on its implications for critical thinking and problem-solving skills (George 2023), as well as concerns around academic integrity and plagiarism detection (Kaushik et al. 2025). Other studies have investigated the perceptions of students and educators regarding the role of generative AI in learning, with findings pointing to both opportunities and anxieties about its integration into teaching and assessment practices (Kaushik et al. 2025; Mahaini 2024). Early work has also examined the ways GenAI reshapes student workflows in programming and data science, from debugging to code explanation (Becker et al. 2023; Korpimies, Laaksonen, and Luukkainen 2024).

When it comes to programming education specifically, generative AI has been explored as a way to help learners produce working code through natural language interaction, potentially lowering barriers for those without prior programming experience. (Ma, Wu, and Koedinger 2023) compared traditional human-human pair programming with "pAIr programming," where learners collaborate with AI coding assistants such as GitHub Copilot. They highlight both benefits (e.g., faster task completion) and challenges (e.g., mismatched expertise, unclear collaboration norms). At the same time, a recurrent theme in the literature is that GenAI is often perceived as a replacement for educators or peers as the first source of help when students encounter difficulties (Holmes 2020; Luckin and Holmes 2016; Kaushik et al. 2025; Mahaini 2024). (Prasad and Sane 2024) argue that because novices can now obtain working solutions with minimal effort, they may bypass essential practices such as planning, monitoring, and reflecting. This shift raises questions about how reliance on AI affects deeper learning, collaboration, and the development of independent problem-solving skills.

GenAI as a Potential No-Code Alternative

Although both no-code AI platforms and generative AI reduce the need for programming expertise, their modes of interaction differ substantially. No-code platforms constrain learners to predefined graphical workflows, ensuring a structured approach but limiting flexibility. Generative AI, in contrast, allows students to express tasks in natural language, producing code that can be executed, tested, and iteratively refined. This affords more flexibility but also requires students to engage with code execution and debugging—skills that may promote deeper understanding of machine learning workflows compared to drag-and-drop interfaces.

Despite the growing interest in both areas, little research has systematically compared generative AI to established no-code platforms in an educational context. Studies of no-code AI highlight their utility for accessibility and democratization (Sundberg and Holmström 2023; Wang and Wang 2024), while research on generative AI focuses largely on programming education and productivity (Ma, Wu, and Koedinger 2023). The question of whether generative AI can function as a no-code alternative for teaching machine learning workflows remains underexplored.

Closest to our work is Sundberg et al. (Sundberg and Holmström 2024), who used a commercial no-code deep learning platform (Peltarion) to let business students train and deploy models through a graphical interface. In contrast, our setting examines generative AI as a potential no-code alternative: while it lowers entry barriers, students still need to run and interpret code, making the interaction less constrained and more akin to programming than to menu-driven platforms.

This study addresses that gap by examining student use of KNIME (as an established no-code baseline) and generative AI (as a new potential alternative). By analyzing students' tool choices, experiences, and outcomes, we provide empirical evidence on the opportunities and challenges of introducing generative AI into workflow-based machine learning education, and reflect on lessons learned for educators considering similar approaches.

Course and Study Design

Course overview

The study was conducted within the AI Essentials module, a polytechnic-wide elective offered to students across diverse diploma programs, including computing, electrical engineering, arts and design, optometry, and maritime engineering. In Singapore, polytechnic education is comparable to community colleges in the United States or Fachhochschulen in Germany, emphasizing industry-oriented, hands-on training at the diploma level. Most participants were in their second or third year of study, and the class brought together students with a wide range of disciplinary backgrounds, making it a particularly suitable setting to explore accessible approaches to AI education.

AI Essentials is a 15-week elective designed to provide students from diverse diploma backgrounds with both a conceptual and practical introduction to artificial intelligence. The module explains what AI is, where it is applied, and its societal implications, with a particular emphasis on machine learning. Core topics include supervised learning (classification and regression), unsupervised learning (clustering), and an introduction to neural networks.

A typical week comprises about one hour of online lesson videos and a three-hour, in-person class. The in-person session is split roughly in half: theory with brief interactive activities in the first half, followed by a hands-on practicum in the second half where students work through the same concepts using either KNIME or GenAI-supported Python workflows. Practical work is not meant to teach a specific tool but to help students build complete workflows that apply machine learning methods to real problems. By the end of the module, students are expected to be able to describe fundamental AI concepts, critically discuss opportunities and limitations, and construct simple ML solutions using accessible platforms. Assessment is distributed across weekly quizzes and assignments, a research-based essay on AI applications, and a semester-long group project.

Practical Sessions and Interventions

The course traditionally includes seven structured practical sessions beginning in Week 2, with no practicals for the first and last chapters. These sessions progressively build core skills in machine learning—data visualization; supervised classification; supervised regression; unsupervised clustering; neural-network-based classification; training a simple neural network; and tuning neural networks. Each KNIME guide combines step-by-step instructions with embedded questions: some check correctness (e.g., "report the model's accuracy" or "what is the R^2 value?"), while others prompt reflection (e.g., "interpret the shape of the loss curve"' or "explain the effects of changing λ").

This semester we introduced a parallel set of GenAI-based guides as the main intervention. These retained the same correctness checks and reflective prompts but replaced detailed click-by-click workflows with higher-level tasks (e.g., "generate code to plot a histogram of feature values"), plus brief environment setup notes (e.g., JupyterLab). KNIME users therefore followed tightly scaffolded workflows, whereas GenAI users prompted, executed, and debugged code as part of the learning process. Students were told that KNIME represented the established baseline; however, they could choose either path, and equivalent instructional and technical support was provided. To ensure a shared foundation, all students completed the Week 2 practical using KNIME before being given a free choice of tools for subsequent weeks.

Instructor Reflection

Choosing KNIME for the shared Week-2 session intentionally reduced early setup friction and gave everyone a common reference point for later comparisons.

During each of the approximately 1–1.5 hours of supervised practice time the lecturer addressed both conceptual and technical issues, including KNIME's deep learning integration and Python library installation.

Instructor Reflection

Running parallel tracks increased live troubleshooting load more than expected, requiring more individualized support during the practical sessions.

In addition, we introduced a motivational GenAI activity in the very first week, before the standard sequence of practicals began. At the start of the opening class, while students were settling in, the lecturer gave a live demonstration: prompting a generative AI tool to produce Python code for a simple image classifier distinguishing between "Labubus" and "Teletubbies," and extending it to use the laptop webcam for live testing.[1] After covering the regular Lesson 1 content, students then attempted a similar activity in pairs, guided only by brief written instructions (see supplementary material). The session concluded with short sharing and reflection, establishing from the outset that the course would combine hands-on experimentation with critical thinking about AI tools.

Instructor Reflection

Incorporating the webcam made the GenAI demo playful and memorable, yet it forced local execution and library installs, creating setup hurdles that left many students struggling; in hindsight, a hosted notebook would have preserved the fun without the overhead.

The primary assessment of practical skills was a semester-long group project, accounting for 20% of the overall module grade. The project required students to develop a credit risk classifier and submit a report documenting their workflow design, modeling decisions, and interpretation of results. Reports also mandated disclosure of any GenAI use for writing assistance, with grading criteria emphasizing reasoning and application over code or KNIME workflow generation.

Research Study

Our study aimed to capture how students engaged with the choice between KNIME and GenAI and what motivated their tool preferences. Data were collected from three complementary sources:

- **Informal feedback:** Ongoing observations and short conversations with students during weekly practical sessions. While not systematically recorded, these interactions provided a broad sense of student struggles and strategies, as nearly all 45 students interacted with the lecturer at some point.
- **Survey:** A short questionnaire administered at the end of the final practical session (before project submission). Of

[1] Labubu (a stylized designer toy) and the Teletubbies (1990s children's TV characters) were chosen as deliberately quirky and not conventionally aesthetic examples, simply to provide visually distinct labels for the demo.

the 44 enrolled students, 30 completed the survey, representing a 68% response rate. The survey focused on tool choice, perceived learning, and experiences with KNIME or GenAI. The full survey is in the supplementary material.

- **Interviews:** Informal, small-group discussions with approximately 10 students during the final consultation session. Conducted with project groups, these conversations offered richer qualitative insights into how students approached their workflows and reflected on tool use.

Participation was voluntary and anonymous; no personally identifiable information or demographic data (e.g., gender, diploma) were collected, in part to protect student privacy given the skewed distribution across diplomas. The study had Institutional Review Board approval, with data stored securely on the lecturer's computer. No incentives were provided.

To avoid grading bias, project grades were excluded from our analysis. Because the use of KNIME or GenAI was obvious in project submissions, incorporating grades could have introduced subjective bias by the lecturer, who was also the assessor. Our design therefore focused on self-reported experiences and qualitative feedback rather than performance metrics.

Survey Evaluation

We received 30 responses to the survey (68% response rate).

Tool choice patterns

The first set of questions focused on students' tool choice. Most respondents primarily used KNIME (n=20), while roughly a third used GenAI (n=9).[2] Table 1 summarizes prior experience and starting confidence (1=very low, 5=very high).

On average, students who chose GenAI reported higher prior programming experience ($\bar{x} = 4.13$, $s = 0.64$) than those who chose KNIME ($\bar{x} = 3.40$, $s = 0.99$). A chi-square test for independence (treating the 5-point programming scale as categorical) did not detect a statistically significant association with tool choice ($\chi^2(4) = 3.74, p = .443$). Given the small sample and ordinal scale, we view this as descriptive rather than confirmatory.

Programming experience was relatively high across the cohort. Although the elective is open polytechnic-wide, most enrolments came from electrical engineering and computing diplomas, with fewer students from non-technical programs (e.g., business, media, arts), which aligns with the generally confident programming self-reports.

In contrast, a chi-square test did find an association between prior experience with GenAI code generation and tool choice ($\chi^2(4) = 14.98$, $p = .0047$): students already familiar with GenAI for coding were far more likely to select it as their primary tool.

[2]One student reported using both tools equally. We retain this case for overall tallies but exclude it from between-group comparisons.

	KNIME	GenAI
Primary tool	20	9
Prior experience with...		
programming	3.40 ± 0.99	4.13 ± 0.64
KNIME	2.50 ± 1.50	2.44 ± 1.23
GenAI (code gen)	1.90 ± 1.25	3.89 ± 0.60
Confidence with chosen tool at start	2.40 ± 1.27	3.89 ± 0.60

Table 1: **Tool choice patterns.** Items used a 5-point scale (1=very low, 5=very high).

	KNIME	GenAI	p
Understanding the ML workflow	3.79 ± 0.79	3.78 ± 0.83	.93
Developing project skills	3.90 ± 0.64	3.67 ± 1.12	.72
Supporting self-paced learning	4.10 ± 0.72	4.33 ± 0.50	.44

Table 2: **Perceived learning and usefulness.** 5-point scale (1=strongly disagree, 5=strongly agree). Mann–Whitney tests indicated no statistically significant between-group differences.

Confidence gains

To gauge change in confidence, the end-of-semester survey asked students to rate their confidence *at the start* and *now* on the same 5-point scale (a retrospective pre–post). A Wilcoxon signed-rank test indicated a statistically significant increase overall ($p < .001$).

We then compared confidence gains between KNIME and GenAI users. A Mann–Whitney U test showed a group difference (U=45.0, p=.029), with KNIME users reporting larger gains—consistent with their lower initial confidence and the course's scaffolding for less experienced users.

Perceived learning and usefulness

Across both tools, students reported similar perceived learning and usefulness (Table 2). These ratings also appeared independent of prior experience with the chosen tool (Table 3), suggesting that both environments were broadly accessible—even for students without much prior familiarity.

Figure 1 summarizes responses to "I would prefer to use this tool in future courses with similar content." Both groups were generally positive (KNIME: median $= 2.5$, IQR $= 2.25$; GenAI: median $= 4.0$, IQR $= 1.0$). After aligning scores to reflect own-tool preference, a Mann–Whitney test showed no statistically significant difference between groups (U=71.5, p=.378).

Influence of prior experience

We did not observe a relationship between prior *programming* experience and change in confidence (Table 4). However, greater prior experience with the *chosen tool* correlated

	Spearman ρ	p
ML workflow usefulness	0.08	.67
Project skills usefulness	−0.16	.40
Self-paced usefulness	0.12	.52

Table 3: **Usefulness vs. prior experience.** No significant relationships between prior experience with the chosen tool and perceived usefulness across outcomes.

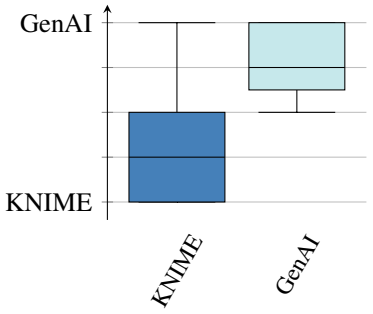

Figure 1: Boxplots for the item "I would prefer to use this tool in future courses with similar content." Scores were aligned to each student's own tool.

with smaller confidence gains, suggesting that increases largely came from learning something new rather than tuning already-familiar skills.

Motivation for tool selection

Across all students, the most common motivations were ease of use (55%) and the visual interface (48%; KNIME-only), followed by speed/efficiency (45%) and familiarity from past courses (38%). Tool-specific patterns were pronounced: KNIME users frequently cited the visual interface (70%), whereas GenAI users emphasized speed/efficiency (89%) and the ability to ask follow-up questions (56%). Ease of use was a major factor for GenAI users (78%) relative to KNIME users (45%). Recommendations by peers were rare (3%). Figure 3 suggests that ease-of-use considerations weighed more heavily for GenAI users than for KNIME users.

Qualitative Student Feedback

Throughout the semester, and in a more focused effort during the final consultation session, we gathered informal feedback from students about their experiences and opinions. This feedback can be grouped into three themes: reasons for choosing a particular tool, perceived learning, and the impact of having two different tools on teaching.

Tool choice

A key question at the start of the semester was whether GenAI-based code generation would be accessible to students without prior coding experience. In practice, most students had at least some coding background, and none of those without coding experience chose to use GenAI. When asked why, the reasons were mainly practical: one student had missed the first lesson, another had not brought a laptop, and both therefore defaulted to KNIME. They added that they would likely have chosen KNIME regardless, seeing it as the safer option; one student explicitly noted they had never used generative AI before.

	Spearman ρ	p
Prior programming experience	−0.17	.37
Prior experience with chosen tool	−0.56	.0015

Table 4: **Prior experience vs. confidence change.** Retrospective pre–post change in confidence was negatively associated with prior *tool* experience (moderate correlation), but not with prior programming experience.

Students also expressed strong preferences about the tool they did *not* choose. For example, a KNIME user said they disliked the debugging that comes with coding, while another voiced doubts about the correctness of AI-generated code. Conversely, one GenAI user recalled from a previous module that "KNIME is horrible." Others cited prior familiarity as their reason for sticking with KNIME. Overall, students were satisfied with their choices and did not report major regrets or frustrations.

Perceived learning

A recurring question among colleagues was what students actually learn when using GenAI beyond copying prompts into ChatGPT. Given the novelty of the technology, many students had as much—or even more—prior experience with it than many lecturers (GenAI users reported an average of 3.8/5 experience with AI-assisted coding; see Table 1).

When asked directly, some GenAI users admitted that parts of the exercises felt like "just copy-pasting" and said they would have liked more explanation of the underlying methods. Interestingly, no KNIME users raised this concern, even though their guides provided no more conceptual background. In practice, both groups achieved the same learning outcomes: they built models and interpreted results. The main difference was that KNIME often required following more detailed procedural steps, while GenAI sometimes felt too easy; consequently, much of the additional learning for both groups appeared to be familiarity with the tool itself rather than deeper methodological insight.

Offering both tools

Offering two tool options directly affected teaching. In previous iterations, when everyone used KNIME, we could control software versions and random seeds so that students obtained comparable results. With GenAI in the mix, this consistency was no longer possible. Whole-class debriefs therefore became less feasible, and we shifted toward more individualized feedback and pair discussions. We initially worried this would disadvantage students.

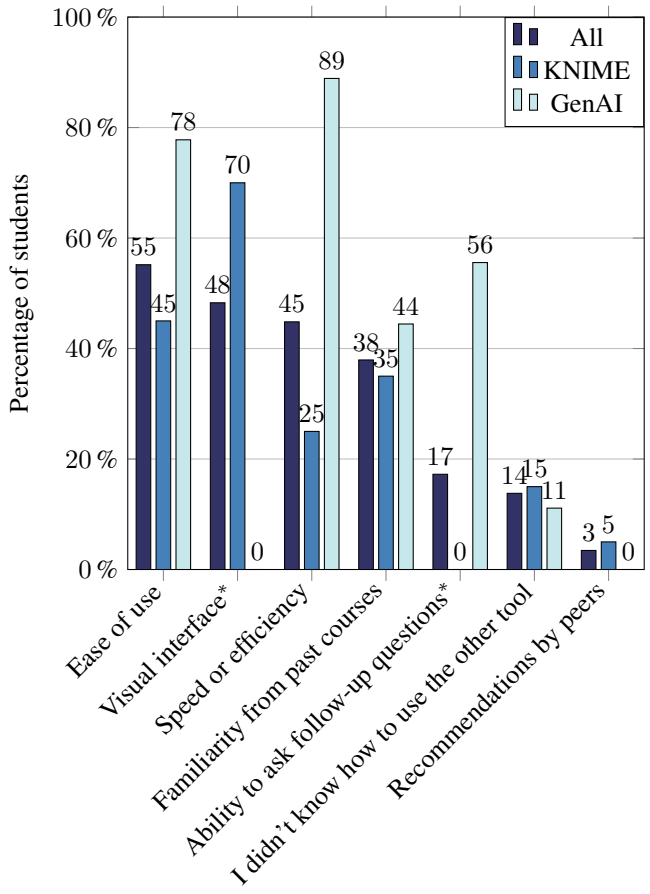

Figure 2: Motivations for tool selection (multiple selections allowed), overall and by tool. Tool-specific items: "Visual interface" (KNIME) and "Ability to ask follow-up questions" (GenAI).

Instructor Reflection

While choice supported autonomy, it also required redesigning wrap-ups to emphasize variability and reproducibility rather than uniform "correct" outcomes.

Students, however, strongly appreciated having a choice. Both informal feedback and the survey indicated support for offering two options: the statement "I believe offering tool choices improves the overall learning experience" received a median agreement of 4/5 from both groups. Even when prompted about potential downsides (less structured whole-class feedback, less uniformity in results), most students reported that the provided guides were sufficient and that they felt confident working independently. A few students did note that differing outcomes between tools (e.g., model accuracies) could be confusing at times.

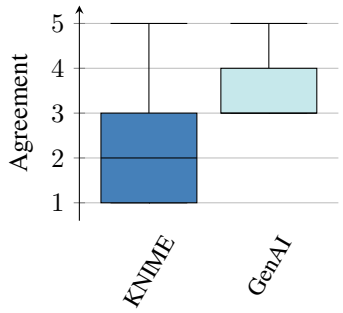

Figure 3: Responses (1=strongly disagree, 5=strongly agree) to "My choice of tool was influenced more by ease of use than by prior programming experience," shown by tool.

Motivational Activity

Students remembered the Week-1 GenAI activity throughout the semester and appreciated the playful idea of generating a webcam classifier. However, the setup proved frustrating: installing Anaconda and packages took considerable time, few students managed to run the demo successfully, and those who did often saw classifiers that predicted only one class. This left several confused, with one student remarking that they "hoped the rest of the course wouldn't be like that."

Discussion

This study explored how polytechnic students engaged with a choice between KNIME and GenAI-supported Python for machine learning practicals. With a relatively small class size ($N = 44$) and a design centered on voluntary tool choice rather than controlled comparisons, our findings are best read as exploratory insights into how students experience and negotiate these options in a natural classroom setting. Within these constraints, several reflections emerged.

Motivations and design considerations

One motivation for offering a GenAI pathway was the mixed experiences with KNIME in earlier course iterations. While students generally liked the tool and evaluated the course positively, some shortcomings persisted. For instance, ensuring consistency across students' results required enforcing specific KNIME versions, which increasingly caused problems: the Keras Deep Learning extension, used in neural network activities, depends on a Python version that cannot be installed on modern MacBooks or on Windows laptops that block unsupported Python releases. Pedagogically, much of the exercise time was consumed by navigation ("open this window, click this button") rather than reflection on why particular steps were taken.

GenAI promised to shift that balance. If students could describe steps in natural language, the system would generate the corresponding code. Yet this raised genuine concerns: could students without coding backgrounds use such workflows effectively, and what would they actually learn if the "how" of coding was outsourced to the model? Our

course, which is designed fot students from diverse backgrounds and emphasizes understanding AI concepts rather than coding proficiency, offered a testbed to examine these questions.

Ideally, one would redesign all practical activities around GenAI and reinvest the time savings into higher-order activities such as evaluation and experimental design, then compare outcomes against a KNIME-based version. In practice, this was not feasible: the module contributes to GPA, the current design is already positively evaluated, and resources for large-scale redesigns are limited. As a compromise, we allowed students to choose freely after completing one shared KNIME session in Week 2, while decoupling that choice from grades. Guides were aligned across tools, differing mainly in the level of procedural scaffolding.

We deliberately avoided integrating specialized AI-enabled IDEs such as Cursor. While such environments offer tight integration of code generation and execution, they represent a specific tool that many students may not be familiar with, and whose availability or interface may change rapidly over time. In contrast, our use of a general-purpose GenAI chatbot reflects a tool that nearly all students already recognize, and whose natural language interface is likely to remain stable across cohorts. This choice also supports reproducibility: course materials and prompts can be reused with different GenAI back-ends, whereas tying the module to a single editor would risk tool obsolescence or vendor lock-in.

Practical insights

The first practical insight concerns the motivational GenAI activity in Week 1. Designed to show that students could train a playful "Labubu vs. Teletubby" classifier within minutes, it instead placed heavy emphasis on setup. Installing Anaconda, configuring JupyterLab, and downloading datasets overshadowed the GenAI component itself. For less confident students, the setup felt daunting, and KNIME appeared simpler and safer. We suspect this early experience contributed to only technically stronger students continuing with GenAI. A streamlined version of this activity is planned for future iterations.

Second, students valued having both options and generally accepted the trade-offs. From a teaching perspective, however, supporting two toolchains widened the range of troubleshooting requests and removed the possibility of fixed "correct" results. With KNIME alone, controlled versions and fixed seeds ensured reproducibility. With GenAI, outputs varied and sometimes contradicted each other. Rather than treating this as a drawback, we see it as an authentic teaching opportunity: it exposes students to the stochastic nature of ML algorithms and raises discussions about reproducibility, robustness, and variability in practice. Stronger reflective wrap-ups at the end of practical sessions would help consolidate these lessons.

Limitations

Several limitations constrain what can be concluded. The study is exploratory, with a modest sample ($N = 44$) and voluntary tool choice, limiting statistical power and introducing self-selection bias. Confidence gains were measured retrospectively rather than longitudinally, and we did not collect demographic information. Prior work suggests that individual differences such as gender, race, and prior experiences can shape the effectiveness of intelligent tutoring systems (Kulik and Fletcher 2016). At the same time, a recent study on LLMs as programming tutors did not find significant performance differences based on gender or race (Lyu et al. 2024). Incorporating demographic and longitudinal data in future work would allow for a more nuanced understanding of who benefits most from different approaches.

Conclusion

Our experience shows that GenAI can be integrated as a viable alternative to established no-code tools in introductory AI education, though student uptake may remain strongest among those with prior coding confidence. Offering both options was well received by students and fostered independence, but increased the complexity of teaching and assessment design.

We recommend a staged, hybrid approach: beginning with a structured, low-barrier tool such as KNIME, while offering GenAI as an optional or parallel pathway, and gradually expanding reflective activities that encourage students to interrogate variability and reproducibility. As a practical next step, instructors considering a full GenAI transition might pilot streamlined activities early in the course, while monitoring how different student groups engage with and learn from the new workflow.

Instructor Reflection

For now, a staged hybrid model—KNIME first, optional GenAI with cloud notebooks and reflection prompts—offers the best balance of accessibility and deeper learning.

Acknowledgments

The authors would like to thank Alan Tsang, Hongyan Chang, Prakhar Ganesh, Ta Duy Ngyuyen, Nguyen Hien Tuan Duy, Tien Chern Chia and Pei Chin Lim for encouragement, helpful discussions and their feedback.

References

Becker, B. A.; Denny, P.; Finnie-Ansley, J.; Luxton-Reilly, A.; Prather, J.; and Santos, E. A. 2023. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 500–506.

Berthold, M. R.; Cebron, N.; Dill, F.; Gabriel, T. R.; Kötter, T.; Meinl, T.; Ohl, P.; Thiel, K.; and Wiswedel, B. 2009. KNIME-the Konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1): 26–31.

Bhattacharyya, S. S.; and Kumar, S. 2023. Study of deployment of "low code no code" applications toward improving digitization of supply chain management. *Journal of Science and Technology Policy Management*, 14(2): 271–287.

Demšar, J.; Curk, T.; Erjavec, A.; Črt Gorup; Hočevar, T.; Milutinovič, M.; Možina, M.; Polajnar, M.; Toplak, M.; Starič, A.; Štajdohar, M.; Umek, L.; Žagar, L.; Žbontar, J.; Žitnik, M.; and Zupan, B. 2013. Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research*, 14: 2349–2353.

George, A. S. 2023. Preparing students for an AI-driven world: Rethinking curriculum and pedagogy in the age of artificial intelligence. *Partners Universal Innovative Research Publication*, 1(2): 112–136.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1): 10–18.

Holmes, W. 2020. Artificial intelligence in education. In *Encyclopedia of education and information technologies*, 88–103. Springer.

How, M.-L.; Chan, Y. J.; Cheah, S.-M.; Khor, A. C.; and Say, E. M. P. 2021. Artificial intelligence for social good in responsible global citizenship education: an inclusive democratized low-code approach. In *Proc. of the 3rd World Conference on Teaching and Education*, 81–89.

Kaushik, A.; Yadav, S.; Browne, A.; Lillis, D.; Williams, D.; Donnell, J. M.; Grant, P.; Kernan, S. C.; Sharma, S.; and Arora, M. 2025. Exploring the impact of generative artificial intelligence in education: A thematic analysis. *arXiv preprint arXiv:2501.10134*.

Korpimies, K.; Laaksonen, A.; and Luukkainen, M. 2024. Unrestricted use of LLMs in a software project course: Student perceptions on learning and impact on course performance. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, 1–7.

Kulik, J. A.; and Fletcher, J. D. 2016. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research*, 86(1): 42–78.

Lethbridge, T. C. 2021. Low-code is often high-code, so we must design low-code platforms to enable proper software engineering. In *International symposium on leveraging applications of formal methods*, 202–212. Springer.

Luckin, R.; and Holmes, W. 2016. Intelligence unleashed: An argument for AI in education.

Luo, Y.; Liang, P.; Wang, C.; Shahin, M.; and Zhan, J. 2021. Characteristics and challenges of low-code development: the practitioners' perspective. In *Proceedings of the 15th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM)*, 1–11.

Lyu, W.; Wang, Y.; Chung, T.; Sun, Y.; and Zhang, Y. 2024. Evaluating the effectiveness of llms in introductory computer science education: A semester-long field study. In *Proceedings of the eleventh ACM conference on learning@ scale*, 63–74.

Ma, Q.; Wu, T.; and Koedinger, K. 2023. Is ai the better programming partner? human-human pair programming vs. human-ai pair programming. *arXiv preprint arXiv:2306.05153*.

Mahaini, D. 2024. *Generative AI in Computer Science Education: A Study on Academic Performance*. B.S. thesis, University of Twente.

Polak, P.; and Anshari, M. 2024. Exploring the multifaceted impacts of artificial intelligence on public organizations, business, and society. *Humanities and Social Sciences Communications*, 11(1): 1–3.

Prasad, P.; and Sane, A. 2024. A self-regulated learning framework using generative AI and its application in CS educational intervention design. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 1070–1076.

Rokis, K.; and Kirikova, M. 2022. Challenges of low-code/no-code software development: A literature review. In *International conference on business informatics research*, 3–17. Springer.

Sahay, A.; Indamutsa, A.; Di Ruscio, D.; and Pierantonio, A. 2020. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. IEEE.

Sundberg, L.; and Holmström, J. 2023. Democratizing artificial intelligence: How no-code AI can leverage machine learning operations. *Business Horizons*, 66(6): 777–788.

Sundberg, L.; and Holmström, J. 2024. Teaching tip: Using no-code AI to teach machine learning in higher education. *Journal of Information Systems Education*, 35(1): 56–66.

Wang, H.-H.; and Wang, C.-H. A. 2024. Teaching design students machine learning to enhance motivation for learning computational thinking skills. *Acta Psychologica*, 251: 104619.

Yan, Z. 2021. The impacts of low/no-code development on digital transformation and software development. *arXiv preprint arXiv:2112.14073*.