Highlights

Process Collapse or Compensatory Strategy? Profiling and Scaffolding Help-Seeking with an AI Tutor

Manuel Valle Torre, Ruben Weijers, Marcus Specht, Catharine Oertel

- Hidden Markov Model trace analysis reveals a universal edit-and-test loop as the dominant mode of AI-supported coding
- Novice delegators achieved equivalent performance to structured coders, but long-term learning effects require further investigation
- Adaptive motivational scaffolding doubled persistence without altering help-seeking behavior
- AI-enabled delegation functions as a compensatory survival strategy, not a failure of self-regulation

Process Collapse or Compensatory Strategy? Profiling and Scaffolding Help-Seeking with an AI Tutor

Manuel Valle Torre[a], Ruben Weijers[a], Marcus Specht[a], Catharine Oertel[a]

[a]*Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628XE, The Netherlands*

Abstract

Generative AI offers immediate educational support but risks fostering "metacognitive offloading," in which learners bypass critical error diagnosis in favor of quick solutions. While concerns about help abuse are widespread, empirical evidence characterizing the behavioral processes of unguided GenAI interaction remains limited. This study addresses this gap by investigating *process collapse*—the breakdown of self-regulated learning cycles—through an exploratory experiment ($N = 40$) involving a diverse cohort of university students and professionals performing data science tasks with an AI tutor. By applying Hidden Markov Models (HMMs) to granular trace data, we identified a pervasive "Tester-Dominant" loop characterized by rapid code iteration. Cluster analysis revealed two distinct archetypes within this pattern: *Structured Coders*, who use the AI for targeted syntax recall, and *Novice Delegators*, who rely more on the AI to get solutions. Pretest data revealed that the first group had higher prior self-reported knowledge in Python and data analysis. However, the performance and engagement of both groups were comparable, suggesting that the AI tutor supported multiple viable learning strategies. Furthermore, we tested an adaptive metacognitive intervention to guide participants towards instrumental help-seeking behavior via conversational nudges. The intervention acted as a safety net: while it did not restructure the underlying process or affect performance, it produced a significant increase in persistence. These findings suggest that for AI-supported novices, "offloading" may be a necessary coping strategy to fill knowledge gaps, and that interventions can have a positive effect by preserving motivation rather than focusing on immediate process optimization.

Keywords: Large Language Models, Help-Seeking Behavior, Metacognitive Scaffolding, Learning Analytics, Hidden Markov Models

1. Introduction

The rapid integration of Generative Artificial Intelligence (GenAI) systems is shifting our relationship with information; reshaping how knowledge is created, accessed, and applied across all sectors of society [1, 2]. Within this transformation, students have emerged as key adopters, integrating Large Language Models (LLMs) like ChatGPT into their academic lives as a constant source of support in their learning process [3, 4]. This adoption is particularly relevant in the post-pandemic landscape, where reduced face-to-face interaction and an emphasis on autonomy place more responsibility on learners to manage their own progress [5].

When confronted with academic challenges, students increasingly utilize GenAI tools as a learning aid, in a process analogous to help-seeking [6]. Effective help-seeking is a cornerstone of Self-Regulated Learning (SRL), requiring learners to monitor their understanding, identify gaps, and assess the assistance they received [7, 8]. However, this process is traditionally fraught with cognitive and social barriers. Novice learners often lack the domain knowledge to articulate the gaps in their knowledge or process, resulting in vague or insufficient queries even with human tutors [9, 10]. Additionally, the fear of appearing incompetent can deter students from seeking help from instructors or peers [11, 12].

LLMs seem to neutralize these barriers. Cognitively, LLMs will answer a question regardless of poor formulation or unclear intent [13]. Socially, they offer a private, non-judgmental resource [14]. While this accessibility promotes their popularity, the "path of least resistance" encourages help abuse—obtaining solutions with minimal effort [15]. This bypasses essential reflection, leading to *metacognitive laziness* and potentially damaging the development of critical thinking skills [16, 17, 18]. Consequently, there is a risk of widening the divide, with skilled learners leveraging LLMs for support while others use them to avoid learning altogether [8].

However, a binary "productive vs. abusive" framing is insufficient for problem-solving learning tasks. A student asking "How do I import a CSV?" is not necessarily avoiding the cognitive work of data analysis; they may simply be acquiring a necessary tool to proceed [19]. Unlike a human tutor, a chatbot may be seen as a replacement for documentation or search engines for such queries, fundamentally changing what constitutes "appro-

priate" help-seeking [20]. Distinguishing such *tool acquisition* queries from true *task offloading* is critical for accurately assessing how students use AI support, as one facilitates the learning process, while the other short-circuits it.

This transforms the effective use of LLMs into a new, essential skill: productive inquiry with LLMs [21, 22]. This emerging skill has the unique potential to be scaffolded in the moment of need by embedding pedagogical guidance directly into the design of the AI agents themselves [23]. By creating systems that guide reflection rather than provide direct answers, we can understand help-seeking in conversational contexts and establish design principles for technologies that foster self-regulation. Ultimately, the goal of the help-seeking process is to advance learning, not merely complete a task [24].

This study aims to understand the nature of productive inquiry with GenAI. While this emerging skill can be scaffolded, doing so effectively depends on a clear definition of what it looks like in an authentic learning context and how traditional help-seeking frameworks apply. Our investigation confronts this definitional challenge directly: the line between constructive, instrumental help (e.g., "How do I import a file?") and executive "help abuse" (e.g., "Give me the code to read the data") becomes blurred when the constructive explanation is a single line of code that is also the quick solution.

To investigate this, we conducted an exploratory two-session study designed to profile and scaffold help-seeking. We situated the experiment within a learning environment featuring programming and data science tasks explicitly designed to leverage AI interaction using the 4-Component Instructional Design (4C/ID) model [25]. This longitudinal design allows us to elicit and contrast the help-seeking patterns that emerge as students tackle new tasks over time [26, 27]. By applying Hidden Markov Models (HMMs) to the resulting trace data, we address the following research questions:

- RQ1: What behavioral patterns characterize student help-seeking in AI-supported data science tasks?
- RQ2: Do distinct behavioral archetypes emerge from these patterns, and how do their learning outcomes compare?
- RQ3: How does adaptive metacognitive scaffolding affect student persistence and engagement compared to unguided support?

2. Related Work

2.1. Theoretical Framework: Self-Regulated Help-Seeking

Academic help-seeking is a fundamental component of self-regulated learning (SRL), understood as a metacognitive cycle involving monitoring, diagnosis, and action [6]. Theoretically, this process consists of five distinct steps: (1) becoming aware of a need for help; (2) deciding to seek help; (3) identifying a potential help source; (4) employing strategies to elicit help; and (5) processing the help received [28]. While conceptually linear, learners often navigate these steps recursively or bypass them entirely under cognitive load [29].

A critical distinction in this process is the learner's intent. **Instrumental help-seeking** involves seeking explanations or hints to enable independent problem-solving, a strategy linked to deep processing and mastery goals [30]. In contrast, **executive help-seeking** is an expedient approach aimed at obtaining a quick answer, often reducing immediate effort but hindering long-term retention [11].

The choice between these strategies is influenced by underlying cognitive traits and prior knowledge [31]. Students with a high *Need for Cognition*—a stable preference for engaging in effortful cognitive tasks—are more likely to employ the processing strategies required for an instrumental approach [32, 33]. Conversely, students experiencing high cognitive load tend toward maladaptive patterns like help abuse or complete help-avoidance [34, 35].

Furthermore, the social context plays a decisive role. Students are more likely to engage in adaptive instrumental strategies when they perceive emotional support and receive explicit praise for asking good questions [36]. This motivational dimension is critical: possessing effective strategies is insufficient if learners lack the motivation to activate them [37]. Indeed, mastery-oriented motivation has been linked not only to deep processing, but also to *persistence* in the face of challenge [38].

2.2. The Disruption: Generative AI and Process Collapse

Generative AI represents a paradigm shift in educational support, and students are increasingly using it as a primary source of assistance [26, 20]. This includes extensive applications, ranging from automated content generation to personalized feedback, identifying unprecedented opportunities to

democratize access to one-on-one tutoring [39]. By offering immediate, high-fluency support, these tools promise to resolve long-standing barriers in scaling educational assistance [3].

However, this technological affordance raises critical questions about the nature of the learning it promotes [4]. By offering immediate, non-judgmental assistance, LLMs alleviate the social and cognitive costs of asking for help [29]. This creates a "convenience trap" that disrupts the self-regulated learning framework: without guidance, students drift toward *metacognitive offloading* [18], relying on the AI to bypass cognitive effort rather than scaffold it [17, 40].

Empirical findings suggest that learners engaging in unguided GenAI interaction can fall into a process collapse, where they bypass critical stages of the help-seeking process (specifically steps 1 (diagnosis) and 5 (processing)), leading to a copy-paste loop [8, 16]. This creates a paradox: the learner receives a rich explanation, but because they have outsourced the metacognitive work of diagnosis and verification, they fail to internalize the information [41]. Furthermore, this passivity leaves students vulnerable to hallucinations, as they lack the deep engagement necessary to critically evaluate the AI's output [26].

2.3. From Static Scaffolding to Dynamic Process Analysis

To address this collapse, we look to instructional design frameworks, primarily 4C/ID, and to Intelligent Tutoring Systems. 4C/ID posits that complex skills require scaffolding that helps learners manage cognitive load during whole-task practice [25]. Managing cognitive load mitigates frustration and abandonment [42], and it may reduce instances in which students ask for a single-step solution rather than decomposing the problem and working progressively [43].

Intelligent Tutoring Systems (ITS) represented the first major effort to automate and scale personalized tutoring [34]. Successful ITS designs implemented this by providing strategic help (e.g., conceptual hints or prompts for self-explanation) rather than direct step-by-step solutions, a method proven to be significantly more effective for deep learning [44]. Classic systems like Betty's Brain demonstrated that adding explicit self-regulation scaffolding and mentor-provided hints not only improved immediate learning outcomes but also prepared students to transfer skills to new contexts [45].

While standard LLMs are not inherently pedagogical, they can function as interactive tutors if constrained to provide this strategic support [46]. For

example, providing students with explicit metacognitive prompts or virtual peer companions while they use GenAI has been shown to improve SRL skills and foster richer reflective engagement [47, 48]. Similarly, targeted pedagogical interventions and high-warmth agent designs can successfully reduce students' executive queries and alleviate the pressure of cognitively demanding tasks [49, 50].

However, effective implementation requires a robust learner model to profile student needs and drive these personalization decisions [51]. Recent implementations such as SRLbot, LLM-integrated AutoTutor, and CodeRunner Agent demonstrate that, when guided by such sound teaching strategies, AI agents can successfully improve engagement and knowledge compared to standard baselines [52, 23, 53].

Finally, to implement adaptive metacognitive support in real-time, we must move beyond static usage metrics to dynamic trace analysis [54]. Self-regulation is a temporal phenomenon, best captured by analyzing the sequence of events (traces) a student generates [55, 56]. Hidden Markov Models (HMMs) have proven effective for this purpose in both structured and Open-Ended Learning Environments (OELEs) [57]. By decoding latent activity patterns from log data—such as monitoring, reflecting, or unproductive wheel-spinning—HMMs allow agents to trigger targeted dialogue strategies precisely when a learner enters a maladaptive phase [58, 59]. The present study adopts this methodology, using HMMs to characterize the process as students learn with AI support.

3. Research Methodology

This study adopts an exploratory approach to characterize help-seeking behaviors in AI-supported learning environments. Our first objective is to diagnose and understand the natural patterns of interaction that emerge when learners navigate complex tasks with an AI tutor (RQ1). Secondarily, we investigate if we can identify groups based on their help-seeking behaviors and if there are differences in performance and persistence (RQ2). Finally, we experiment with an adaptive motivational scaffolding and the subsequent impact on behavior, performance, and persistence (RQ3). We first describe the pilots that informed our exploratory focus and the operationalization of student behavior.

Running Example

To make the interaction between components more concrete, we will use a running example of two hypothetical participants, Alex and Sam.

3.1. Pilot Study

We conducted two pilot studies to calibrate the system and the tasks.

Pilot 1 (High School Students - Introductory Python): The first pilot involved 18 high school students learning introductory Python through regular lectures, followed by work sessions on a Jupyter environment with an AI tutor. Students had extensive code skeletons in the notebooks, together with instructions and conceptual information. The AI tutor used Llama3.1:70b, prompted as a Socratic tutor (see Appendix), to answer questions and provide guidance. During the 12 weeks of work sessions, students were free to interact with the LLM AI tutor for support. This pilot revealed a natural tendency toward executive help-seeking: students frequently asked "fix this" or "what should I do?" rather than seeking to understand the underlying concepts. This tendency persisted until the teacher's explicit pedagogical intervention reversed it [60], suggesting that instrumental help-seeking behavior can be influenced by metacognitive scaffolding. Since students lacked basic knowledge, they often omitted critical context or asked vague questions, resulting in generic responses.

Pilot 2 (Graduate Students - Data Science Tasks): In a second pilot, we tested a version of the current data science tasks with 7 graduate students with low or no programming experience. These tasks used simple guiding questions without conceptual explanations or code scaffolding. The AI tutor was powered by Gemma2:27b and also served as a Socratic tutor (see Appendix) to answer questions and provide guidance specifically for data science tasks. To eliminate noise caused by poor prompting (observed in Pilot 1), we enhanced the AI tutor's access to context, such as user code and errors, and task objectives. In a few cases, the participant's question was buried among the error logs and chat history, leading the AI tutor in circles instead of answering. Participants became unmotivated and abandoned tasks early.

Implications for Main Study Design: Based on these pilots, we implemented several key design improvements:

1. **Multi-Agent Architecture:** We developed a multi-agent system (described in detail in Section 3.5) consisting of an expert and a tutor

agent. This architecture enables the real-time generation of adaptive responses and improves the quality when using local LLMs [61]. Furthermore, it allows us to manage the information that the LLM receives in each step, mitigating the risk of getting lost, as in pilot 2.

2. **Task Scaffolding:** Adopting the 4C/ID framework ensured that observed behaviors were due to learner strategy, not insurmountable task difficulty.
3. **Adaptive Intervention Design:** The system should be able to detect conditions and trigger responses in real time. This enables us to evaluate if the change in behavior observed in Pilot 1 could be shifted by the agent itself.

3.2. Experimental Design and Procedure

This study was designed as an exploratory investigation into student help-seeking behaviors with AI tutors. We employed a two-session design to assess the stability of help-seeking strategies over time and participants' willingness to return for the second session [26, 27].

- **Session 1 (Exploration)**: Participants engaged with introductory data science tasks (Tasks 1-2).
- **Session 2 (Persistence)**: A follow-up session (Tasks 3-4) two weeks later served to measure voluntary re-engagement and the stability of strategies.

Participants were instructed to engage for at least 1 hour per session, but there was no automatic cutoff. The sessions were separated by approximately two weeks, they took place online, unsupervised, with participants keeping their own time. For analysis, we defined a "work-sprint" as the first continuous period of activity before any inactivity gap exceeding 30 minutes, ensuring we captured focused engagement rather than sporadic check-ins.

Running Example

Alex and Sam are two participants in the study.

3.3. Participants

Participants were recruited through open invitations in the institution's forums, mentoring courses for new bachelor students, and mailing lists for individuals interested in participating in research studies. This yielded a diverse cohort of university students and working professionals. Participation was voluntary and driven by a genuine interest in learning; the study was not part of any mandatory coursework. The invitation offered them the opportunity to learn the basics of data science, receive instant AI feedback, and a 15-euro gift card for their time. The reward was not contingent on completing the tasks, but rather on working for 1 hour in each of the two sessions.

From an initial pool of 45 participants, we derived an analytic sample of $\mathbf{N = 40}$ using engagement filters. These filters removed participants who opted out, those who did not use the system as designed, and those with minimal engagement (less than 10 minutes of interaction). This sample size aligns with similar exploratory studies examining student-AI interaction patterns [43, 62, 63, 47]. The study was approved by our institution's Ethics Committee.

Running Example

Alex, a university student, consents to the study. After completing the questionnaire, they log in to the system and start session 1. Sam, a project manager, follows the same procedure.

3.4. Materials: JELAI and Data Science Tasks

The experiment was conducted within **JELAI** (Jupyter Environment for Learning Analytics with AI), an open-source, multi-agent LLM tutor system operating in a JupyterHub environment [64]. The system captures fine-grained trace data from all student activities. These logs can be used by the system for contextual information or to infer underlying cognitive strategies during analysis [56]. Figure 1 shows the student-facing interface.

For this study, JELAI's architecture consisted of multiple agents, mainly an expert and a tutor, shown in Figure 2 (left). This Mixture of Agents (MoA) architecture enabled us to improve the performance of the open-source LLMs we run locally [61]. This ensures that all data and interactions remain within our servers.

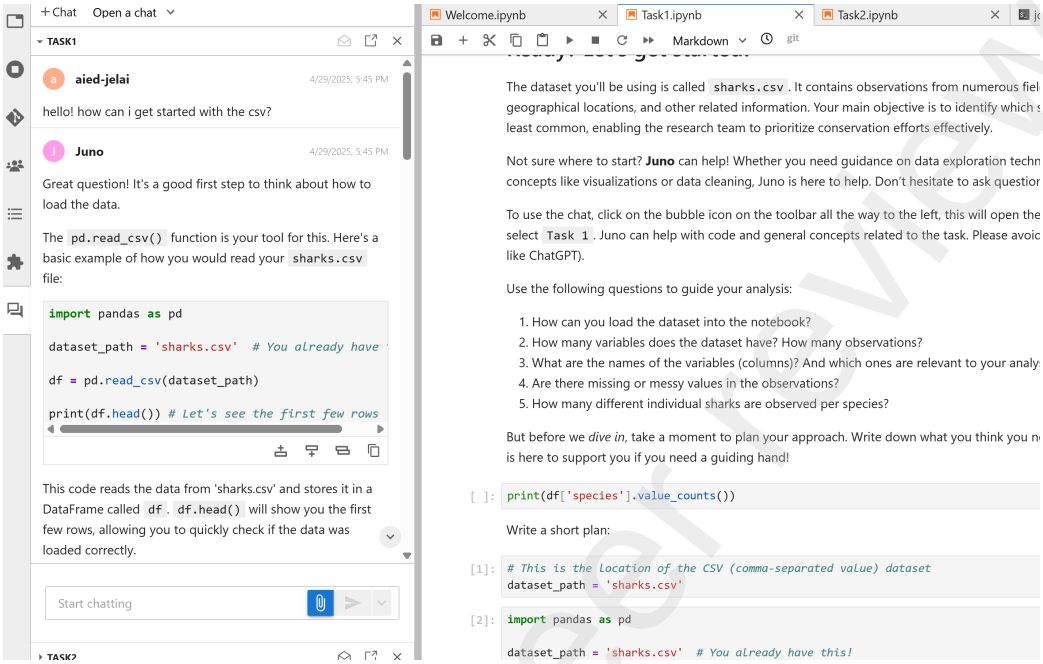

Figure 1: The JELAI interface showing a student interaction with the AI tutor Juno. Left: chat panel with conversation history. Right: Jupyter notebook with task instructions and code cells.

The *expert* agent focuses solely on data science and Python programming. It processes the student's message and the technical context: executed code, error messages, task description, and learning objectives to generate a factual answer. The *tutor* agent integrates the conversation history with expert information to generate an appropriate answer for the student. The tutor is instructed to provide short and concise answers, offering guidance rather than full solutions[1]. The classifier component in the figure is described below in Section 3.5, but it does not affect the system in this stage. The workflow was powered by Gemma3 (4B for the expert and 27B for the tutor). Responses took 5-6 seconds to complete; students saw status messages (e.g., 'Let me think for a second') during processing.

The **learning tasks** were designed following the 4C/ID framework [25], which structures learning around whole-task practice with decreasing sup-

[1]The full prompt will be added after submission.

port. The core of 4C/ID comprises a series of *learning tasks* that form the backbone of the educational experience. The conceptual information and progression of the tasks were in line with Data Science education materials [65, 66]. In this study, there were four data science tasks of increasing complexity, centered on an exploratory data analysis of a realistic "shark scenario" dataset:

- **Session 1 Tasks:** Task 1 (species inventory) and Task 2 (Physical characteristics).
- **Session 2 Tasks:** Task 3 (geolocation analysis) and Task 4 (Movement patterns).

According to the 4C/ID model, these tasks are supported by other components. The *procedural information* included small exercises, with demonstrations and code skeletons provided in the notebooks to reduce the cognitive load of routine syntax recall. The *supportive information* was integrated with conceptual explanations and examples to help learners build mental models (germane load). The AI tutor was designed to complement both, with students requesting further information as needed. The *part-task practice* was achieved by reproducing learned procedures within and across tasks. By providing structured code skeletons and clear sub-goals, this design aims to minimize the initial difficulty that may drive novices to delegate entire tasks to AI, encouraging them instead to engage with specific sub-problems.

Running Example

Both Alex and Sam start session 1 in the JELAI environment. As a 4C/ID learning task, it begins with a scenario: they are junior marine biologists working with a dataset of shark sightings. The first task involves inventorying individuals per species, starting with a brief explanation of Pandas.

3.5. Experimental Probe: Adaptive Metacognitive Scaffolding

For a subset of participants (assigned randomly during registration), we activated a motivational intervention. The system identifies when participants ask an executive question followed by an instrumental question, and it positively reinforces the student for asking the instrumental question:

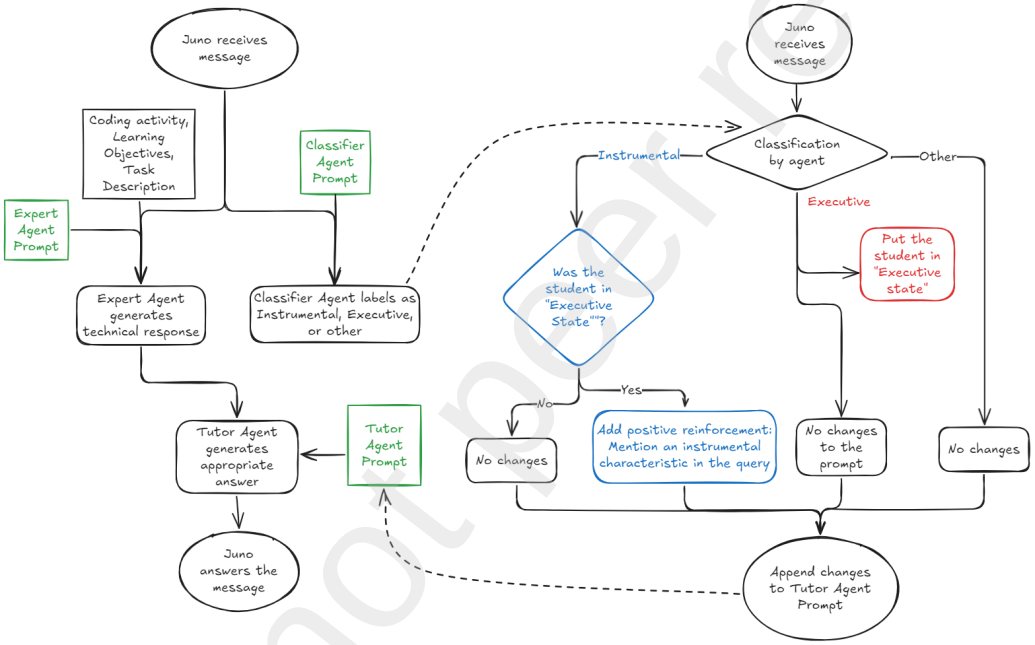

Figure 2: JELAI system workflow. Left: General agent pipeline where Juno receives a message, classifies help-seeking intent, generates expert content, and produces a tutoring response. Right: Adaptive intervention logic, showing positive reinforcement for instrumental queries.

"That's a great way to formulate a question to understand the content," or "This last question was a good way to approach the problem."

We use a classifier (Figure 2) as a third agent to identify the student's intent The *classifier* agent was developed using the MIPROv2 optimizer[2] from the Declarative Self-improving Python (DSPy) framework to categorize student queries according to their Intent (instrumental, executive, or other). The classifier was trained and tested on data from our Pilot 1 with 86% accuracy.

This intervention was based on the adaptive tutoring agent logic displayed in Figure 2 (right side of the figure).

1. **Help-Seeking Classification:** Upon receiving a student message, an LLM-classifier first categorizes it as instrumental, executive, or other (greetings, unrelated, answer to a follow-up question by Juno).
2. **Adaptive Response Logic:** Juno's response is shaped by this classification. A shift from executive to instrumental behavior is met with explicit positive reinforcement. This design choice is grounded in research showing that perceived support and praise for good questions are linked to greater achievement gains and encourage the very behaviors we aim to foster [38].

Everything else remains the same, including the base agents, system prompts, and task knowledge base.

Running Example

After importing the data, both Alex and Sam encounter a `KeyError` when filtering species. Both type "fix this error" to their AI tutor, an executive query. The AI tutor responds to each one: "You're getting a KeyError because Pandas is case-sensitive..." Sam moves on to the next step, while Alex asks "why is it called a key error?" The AI tutor responds "It's great that you want to understand the reasoning behind it! In Pandas..."

[2] https://dspy.ai/api/optimizers/MIPROv2/

3.6. Measures and Analysis

3.6.1. Intent Classification

For help-seeking **intent**, we developed a **five-class pedagogical taxonomy** using a classifier with a 2-turn context window:

- **Tool Acquisition:** Queries requesting specific syntax or library functions.
- **Instrumental:** Questions aimed at deepening conceptual understanding.
- **Executive:** High-level requests for task completion without conceptual engagement.
- **Response to Bot:** Direct responses to the AI tutor's scaffolding prompts.
- **Other:** Social interaction or off-topic chat.

These categories emerged via manual labelling of 398 messages by 2 coders, with disagreements resolved through discussion. These messages were divided as 70% for training, 20% for testing and 10% for evaluation of the classifier (Weighted F1=0.83, $\kappa = 0.78$).

3.7. Behavioral Process Modeling (RQ1)

For the help-seeking *process*, we operationalized behavioral patterns as sequences of trace log events (errors, queries, code edits) [29, 8]. For each participant's study session, we identified the first continuous "work-sprint" and analyzed activity within each task separately.

We trained a single unified HMM on a 12-dimensional observation vector for each event, utilizing all data to learn a shared state representation. The feature vector included standardized measures of:

- **Mechanical Action:** Binary flags for error generation, code editing, and code execution.
- **Intent Class:** One-hot encoding of the 5-class taxonomy.
- **Paste Source:** Binary flags for internal code copy-paste, copy-paste from AI-chat, or pasting content from external sources. We use clipboard logs to infer the source of paste events.

- **Temporal Dynamics:** Log-transformed time since the previous event.

We used Gaussian HMMs with diagonal covariance, trained with the Baum-Welch algorithm (50 iterations, random initialization with seed=42), consistent with prior work on modeling latent behaviors in open-ended learning environments [57, 58]. This approach treats features as conditionally independent given the hidden state, which is appropriate for our mixed feature types. We determined the optimal number of states using the Bayesian Information Criterion (BIC) on the full dataset.

3.8. Behavioral Clustering (RQ2)

To identify latent behavioral archetypes, we applied K-means clustering to the participants' HMM state-occupancy vectors. To focus specifically on interaction strategies rather than coding proficiency, we filtered the feature vectors to include only the occupancy of "Help-Seeking" states (e.g., delegating, asking for tools, responding to questions), excluding the dominant "Code-Manipulation" states (testing, debugging, reorganizing code), which are common to all users. We determined the optimal number of clusters based on the Silhouette Score and interpretability of the resulting profiles. We compared the archetype profiles using Welch's t-tests for continuous variables, considering the unequal sample sizes.

3.8.1. Pre-Measures and Baseline Equivalence

The **pre-measures** include basic demographics, self-reported familiarity with Python and data analysis to gauge prior knowledge, and the short form of the Need for Cognition scale (NCS-6) to assess intrinsic motivation for cognitive effort [33].

3.8.2. Performance Measures

Subtask completion is assessed across 42 skills distributed over the four tasks. Scoring was automated via regex patterns matching expected outputs (validated against trace logs):

- **Not Attempted (0 pts):** No evidence in logs.
- **Attempted (1 pt):** Valid attempts (code/cell edits) but no successful solution.
- **Completed (2 pts):** Successful code execution matching solution patterns.

We report two metrics derived from this scoring: *Total Points* (sum of all scores, including partial credit for attempts) and *Completion Rate* (percentage of subtasks where the full 2 points were awarded).

For **Persistence**, we analyze the attrition rate as the proportion of participants returning for Session 2 using Fisher's exact test to compare between groups. We compared the archetype profiles using Welch's t-tests for continuous variables and Chi-squared tests for categorical variables.

3.8.3. Experimental Probe Evaluation (RQ3)

For participants who experienced the experimental intervention, we compared them to the rest of the population using all pre-measures and performance measures described above.

Running Example

The system logs Alex's and Sam's messages, classifying each as "executive" or "instrumental." Their interaction sequences (error → query → edit → success) become HMM observations. Alex returned for Session 2, but Sam did not, contributing to persistence statistics.

4. Results

This section presents the results of our experiment, organized by research question. The analysis is based on an analytic sample of $N = 40$ participants, using the first work-sprint per session as described in Section 3.6. The trace data and classified messages are published in [60].

4.1. RQ1: Diagnosis - Baseline Strategies

Analysis of the full cohort ($N = 40$ students, $N_{tasks} = 105$ task-segments) reveals the natural problem-solving strategies of students in open-ended data science tasks. These task-segments include all activities per task within the first continuous work-sprint of each participant: if a student worked on tasks 1 and 2 during their first work sprint, they generated two task-segments.

4.1.1. Identified Behavioral States

The HMM trained on the full dataset ($N = 11,925$ events) identified 9 latent behavioral states. The Bayesian Information Criterion (BIC) confirmed this as the optimal configuration for granular yet interpretable separation:

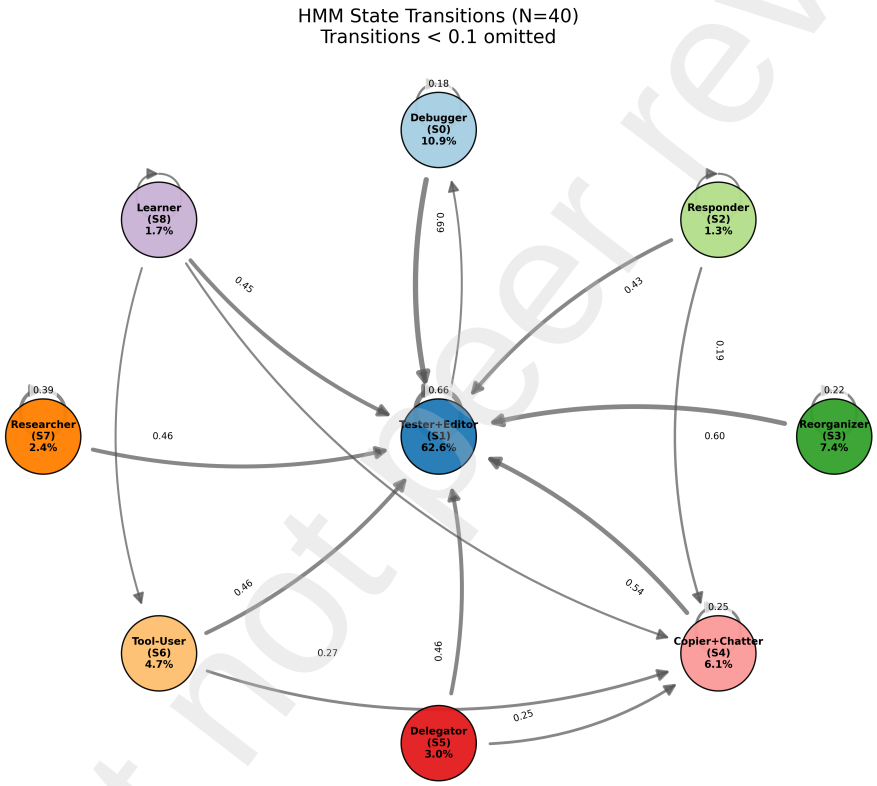

Figure 3: Occupancy and Transition Network of the Hidden Markov Model (N=40 participants)

- **State 0 (Debugger) - 10.9%**: Dominated by error events (+2.9σ). Represents struggling with code execution.
- **State 1 (Tester+Editor) 62.6%**: The most frequent state, characterizing the standard loop of editing and running code.
- **State 2 (Responder) - 1.3%**: Direct interaction with the AI tutor (+8.8σ Juno). Represents engaged dialogue.
- **State 3 (Reorganizer) - 7.4%**: Internal code pasting (+3.5σ). Represents rearranging code blocks.
- **State 4 (Copier+Chatter) - 6.1%**: Pasting AI code (+3.5σ) and off-topic chat (+1.7σ). Represents accepting help and possibly thanks or confirmation.
- **State 5 (Delegator) - 3.0%**: Executive commands (+5.7σ). High-level requests without conceptual engagement.
- **State 6 (Tool-User) - 4.7%**: Tool acquisition queries (+4.5σ). Seeking syntax or procedural help.
- **State 7 (Researcher/Cheater) - 2.4%**: External pasting (+6.4σ). Bringing in code from outside the environment.
- **State 8 (Learner) - 1.7%**: Instrumental queries (+7.7σ). Active conceptual engagement.

4.1.2. Dominant Behavioral Patterns

The occupancy rates and transitions reveal that the "Tester+Editor" state acts as a gravitational center for all students, accounting for the majority of session time, in the center of Figure 3. Transitions out of this state typically lead to "Debugger" (when errors occur) or "Tool-User" (when syntax help is needed), before returning to the coding loop. Notably, the "Delegator" state (executive help-seeking) often transitions to "Copier+Chatter" or directly back to "Tester," confirming the *request code* $\rightarrow$ *paste* $\rightarrow$ *run* usage pattern.

4.2. RQ2: Behavioral Archetypes and Outcomes

Using the clustering methodology described in Section 3.8, we identified two distinct behavioral archetypes:

- **Cluster 0: The Delegator** ($N = 13$): Characterized by lower coding autonomy and higher reliance on AI dialogue.
- **Cluster 1: The Structured Coder** ($N = 27$): Characterized by higher autonomy, systematic testing, and specific tool inquiries.

Table 1: HMM State Occupancy by Cluster (Session 1)

State Label	Cluster 0	Cluster 1	Diff	p-value
Tester+Editor	52.9%	68.3%	-15.4%	0.001
Debugger	6.3%	12.2%	-5.9%	0.003
Copier+Chatter	14.6%	3.2%	+11.5%	0.002
Delegator	5.2%	1.7%	+3.5%	0.008
Tool-User	7.0%	4.1%	+2.9%	0.022
Reorganizer	5.6%	6.4%	-0.8%	0.490

Table 2: Help-Seeking Intent Distribution by Cluster

Intent Category	Cluster 0	Cluster 1	Diff	p-value
Executive	30.1%	27.9%	+2.2%	0.74
Tool Acquisition	31.9%	42.3%	-10.4%	0.62
Instrumental	13.4%	13.9%	-0.5%	0.64
Answer Juno	13.2%	8.5%	+4.7%	0.90
Other	11.4%	7.3%	+4.1%	0.25

4.2.1. Archetype Intent and Behavior Differences

The divergence in strategy is best captured by the HMM State Occupancy profiles (Table 1), which reveal significant differences in problem-solving mechanics. Cluster 0 is defined by significantly higher occupancy in passive states: `Copier+Chatter` and `Delegator`, confirming a strategy of delegation and code copying. Cluster 1, conversely, is *Tester+Editor* dominant and

spends more time in the `Debugger` state, reflecting a strategy of active construction and self-correction with some trial-and-error. While differences in classified Intent (Table 2) follow similar trends (higher Tool Acquisition for Cluster 1), the high variance renders them statistically indistinct, suggesting that the *process* (HMM states) differentiates these users more reliably than the frequencies of individual queries.

4.2.2. Archetype Profiling

To understand the composition of these archetypes, we compared the demographic and pre-test characteristics of the students in each cluster (Table 3). A clear distinction emerged based on prior experience: Cluster 1 (Structured Coder) students had significantly higher self-reported Python experience compared to Cluster 0 and significantly higher general data analysis experience. Notably, while there were visible differences in Gender (53.8% vs 33.3%) and Education (30.8% vs 59.3%), these did not reach statistical significance according to Chi-squared tests, likely due to the smaller sample size of Cluster 0 ($N = 13$). There were also no significant differences in Age, Motivation, or Need for Cognition. This confirms that the behavioral archetypes capture a "Novice vs. Intermediate" split, where novices (Cluster 0) compensate for low prior knowledge by relying on the AI for code generation (Executive help-seeking), while intermediates (Cluster 1) use the AI for targeted syntax recall (Tool Acquisition).

Table 3: Demographic Profile by Behavioral Archetype

Variable	**C0 - Delegators**	**C1 - Coders**	**p-value**
Continuous			
Python Experience (1-5)	1.83 ($SD = 0.83$)	2.85 ($SD = 1.06$)	**0.003**
Data Analysis Exp (1-5)	1.33 ($SD = 0.65$)	2.56 ($SD = 1.12$)	**<0.001**
Age (1:16-24, ... 5:55+)	2.08 ($SD = 1.51$)	2.30 ($SD = 1.44$)	0.684
Motivation (1-5)	2.67 ($SD = 0.89$)	2.41 ($SD = 0.89$)	0.409
Need for Cognition	3.29 ($SD = 0.50$)	3.30 ($SD = 0.41$)	0.978
Categorical			
Gender (% Female)	53.8%	33.3%	0.305
Education (Bachelor+)	30.8%	59.3%	0.337

4.2.3. Arechetype Engagement and Outcomes

Table 4 compares the engagement and performance metrics between the two archetypes across both sessions.

Despite their divergent strategies, both groups invested similar time in Session 1 and achieved statistically equivalent learning outcomes. This suggests a phenomenon of *equifinality*: the "Novice Delegator" strategy of delegating code generation to the AI was nearly as effective for task completion as the "Structured Coder" strategy of manual coding and debugging, at least in the short term. However, the behavioral mechanisms differed fundamentally. As shown in Table 1, Cluster 0 relied significantly more on `Copier+Chatter` and `Delegator` states, while Cluster 1 dominated the `Tester+Editor`.

Longitudinally, a divergence in strategic adaptation emerged. Cluster 0 displayed behavioral stability: among returning participants ($N = 6$ pairs), there were no significant shifts in state occupancy between sessions, suggesting they maintained their delegation strategy even as tasks became more complex. In contrast, Cluster 1 ($N = 14$ pairs) demonstrated significant adaptation. In Session 2, they significantly reduced their time in the `Tester+Editor` state and increased their reliance on `Delegation` and `Copying`. This suggests that capable learners adaptively shifted towards AI-offloading strategies to manage increased cognitive load, whereas novices (still novices) remained in their initial pattern.

Persistence was also similar between the two clusters. The return rate for Session 2 was 46.2% (6/13) for Cluster 0 and 51.9% (14/27) for Cluster 1. This demonstrates that the "Novice Delegation" strategy is not inherently less sustainable or more frustrating than the "Structured Coder" strategy. Both pathways are equally viable in an LLM-supported environment.

4.3. RQ3: Intervention Mechanism and Persistence

To understand whether the intervention influenced student behavior, we analyzed the mechanism of "Positive Reinforcement." This mechanism was triggered for selected participants when they exhibited the eligible behavior: upon exiting an "Executive" help-seeking state by asking an "Instrumental" question, the system provided a positive metacognitive scaffold (reinforcement). The intervention typically triggered 26.1 minutes (median) into the session, serving as an early-to-mid session reinforcement of effective behavior.

Table 4: Outcomes by Behavioral Archetype (Session 1 & 2)

Metric	C0 - Delegators	C1 - Coders	p-value
Session 1			
Points	23.9 ($SD = 8.0$)	28.3 ($SD = 5.5$)	0.09
Completion Rate	56.3%	63.6%	0.15
Time on Task (min)	62.8	105.2	0.40
Executive Intent (%)	12.5%	8.5%	0.02*
Instrumental Intent (%)	57.4%	73.4%	< 0.001**
Session 2 (Persisters)			
Points	17.0 ($SD = 10.3$)	18.9 ($SD = 8.9$)	0.70
Completion Rate	68.5%	72.6%	0.80
Time on Task (min)	52.4	78.2	0.15
Executive Intent (%)	5.9%	12.2%	0.09
Instrumental Intent (%)	69.5%	62.9%	0.24
Rate (Returning for S2)	46.2% (6/13)	51.9% (14/27)	1.00

4.3.1. Impact on Help-Seeking Intent

We examined whether receiving the intervention altered the overall help-seeking intent of the participants. Comparing the help-seeking intent distribution of participants who received the reinforcement ($N = 12$) against the non-reinforced population ($N = 28$), we found no significant differences in strategy. The Reinforced group was statistically indistinguishable from the general population in terms of Executive intent (24.6% vs 25.3%, $p = 0.89$) and Instrumental intent (17.1% vs 12.7%, $p = 0.22$). The "Tool Acquisition" intent also showed no significant difference (34.3% vs 46.0%, $p = 0.12$). Thus, the intervention did not shift the fundamental help-seeking profile of the users.

4.3.2. Impact on Performance and Engagement

In Session 1, participants who received reinforcement achieved statistically equivalent scores ($M = 28.2$ vs $M = 26.2$, $p = 0.40$) and completion rates (64.8% vs 59.6%, $p = 0.28$) compared to the non-reinforced population. However, they were significantly more engaged with the AI chat interface, asking roughly twice as many questions ($M = 41.1$ vs $M = 21.4$, $p = 0.012$). While Reinforced students spent more time on the task (Median 141 min vs 96 min), this difference was not statistically significant due to high variance.

This gap in message volume disappeared in Session 2. Among persisting

students, there were no significant differences in the number of questions asked ($M = 18.5$ vs $M = 18.2$, $p = 0.94$), time on task ($p = 0.56$), or performance metrics ($p = 0.53$) between the groups.

4.3.3. Impact on Persistence

The most significant effect of the intervention was on participant retention.To understand the mechanism, we analyzed how many participants actually engaged in the target behavior (Executive → Instrumental shift).

In summary, approximately 35% of the population ($N = 14$) did not trigger the intervention condition, typically by avoiding Executive help-seeking. From those 14, 8 were in the intervention-eligible group, and 6 in the non-eligible group. For the remaining 65% ($N = 26$) who *did* exhibit the target behavior, the intervention defined the outcome:

- **Reinforced ($N = 12$):** Eligible participants received praise and persisted at **83.3%** (10/12).
- **Ignored ($N = 14$):** Non-eligible participants showed the exact same adaptive behavior but received no feedback; only **42.9%** (6/14) returned.

This demonstrates that while the behavior itself (questions and strategies) was equally common in both groups, the *validation* of that behavior was the decisive factor for retention.

Table 5: Persistence Rates by Interaction Depth

Group	**N**	**Persisted (S2)**	**Rate**
Reinforced (Eligible, Triggered)	12	10	83.3%
Not-reinforced (Eligible, Untriggered)	8	4	50.0%
Not Eligible (All)	20	6	30.0%
Total	40	20	50.0%

Comparison: Reinforced vs Not Eligible ($p = 0.005$, OR=11.6)

Moreover, this *safety net* effect was universal across behavioral archetypes. The intervention did not influence cluster formation (Independence χ^2, $p > 0.05$) and was equally effective for both groups (Interaction $p = 0.48$). Notably, while the Novice Delegators in the non-reinforced group experienced

low persistence (16.7%), those who received reinforcement persisted at the same high rate (83.3%) as the Structured Coders. The intervention thus effectively closed the retention gap between struggling novices and experienced students.

5. Discussion

This study aimed to characterize student help-seeking behaviors and find discernible patterns in AI-supported programming tasks. We also sought to determine if an adaptive intervention could influence these patterns. Our findings reveal a nuanced reality: while all students gravitate toward incremental coding cycles (RQ1), novices successfully employed delegation as a compensatory strategy to achieve equifinal outcomes to the more experienced participants (RQ2). Regarding the intervention, while it did not alter these underlying strategies, it acted as a safety net that significantly increased learner persistence (RQ3).

5.1. The Universality of the Tester Loop (RQ1)

Our diagnosis of the learner behaviors (RQ1) reveals that in an open-ended coding environment, the `Tester+Editor` loop is the center of all student activity. All participants spent the majority of their time in this state, confirming that the primary mode of interaction with the AI tutor in this context is not conversation but iterative co-construction [67]. On the other hand, the prominent transition from `Delegator` state (asking for code) directly to `Tester` (running it) may represent a form of *metacognitive offloading*, where the assessment of the received help is missing [16].

Furthermore, a tension emerged between the instructional design and the AI's affordances. While the 4C/ID task structure successfully restricted the *scope* of help-seeking to specific sub-problems (preventing full-task offloading), the LLM encouraged shallow *depth* of processing within those subtasks [25]. Instead of engaging in conceptual diagnosis and understanding, students treated the AI as a syntax dictionary or a debugger, entering iterative cycles of tool acquisition and coding that may prioritize immediate function over understanding. Because novices frequently struggle to formulate the exploratory, instrumental prompts needed for conceptual learning [40], they default to these fragmented, executive queries to rapidly offload the cognitive burden of syntax recall and error diagnosis [8].

5.2. Two Paths to the Same Outcome (RQ2)

A key finding of this study is the *equifinality* of the observed strategies. Despite the radically different approaches of the Novice Delegator and the Structured Coder, both groups achieved statistically equivalent performance outcomes in terms of points and completion rates. This challenges the binary of "productive vs. abusive" help-seeking [34] in the context of learning with Generative AI. For novices, the `Delegator` state (asking for code/fixes) may not represent a failure of learning, but a functional compensatory strategy that allowed them to perform at the same level as their more experienced peers.

Crucially, this equivalence was not driven by cognitive disposition—both clusters exhibited virtually identical Need for Cognition scores and similar task engagement—ruling out a "lazy student" interpretation. Our results confirm that prior knowledge drives the choice of strategy [31]: novices gravitated toward delegation while intermediates used targeted tool acquisition. In traditional frameworks, this executive pattern is viewed as maladaptive because it bypasses the cognitive effort required for schema construction [11], and the model predicts it should lead to worse outcomes.

Rather than a failure of self-regulation, however, these participants employed delegation as a *compensatory survival strategy*. Lacking the syntactic fluency to construct code from scratch or ask for specific syntax, they utilized the AI to bridge the gap between their conceptual intent and valid Python syntax. This aligns with recent findings on metacognitive offloading [8], but adds a critical nuance: for novices without prior programming knowledge, this offloading may be the only mechanism that prevents the friction of syntax errors from overwhelming their cognitive load and driving abandonment [34]. Our data suggest that, in the AI-supported context, the AI disrupts the presumed causal chain between help-seeking type and learning outcomes. Delegation produced equifinal performance, not the degraded outcomes the traditional model predicts.

This performance equivalence should not be confused with learning equivalence. While the outcomes were comparable, the process data reveal a risk of stagnation. Longitudinal analysis showed that Structured Coders adaptively shifted their strategies in Session 2 to manage increased task complexity, while Novice Delegators remained in their delegation loops. This suggests that while AI-enabled delegation is an effective scaffold for immediate performance—allowing novices to produce results comparable to their more experienced peers—it risks creating a *competence illusion*: the user

feels capable but remains entirely dependent on the tool [18]. Whether this delegation strategy can serve as a stepping stone toward independent mastery, or whether it entrenches dependence, remains an open question that requires longitudinal investigation beyond the scope of this study.

5.3. Scaffolding as a Persistence Safety Net (RQ3)

While the adaptive intervention did not change participants' process or help-seeking strategies, it had a profound impact on their persistence in the learning environment (RQ3). The implementation of a "positive reinforcement" for instrumental questions acted as an **agency-preserving safety net**. By offering strategic prompts integrated into the conversation [44], the system likely managed the frustration that leads to help avoidance [34]. Crucially, this effect was achieved without a significant change in performance scores [48], highlighting the content-independent value of motivational scaffolding [38].

5.4. Implications for AI in Education

Our findings carry several implications for the design of AI-supported learning environments. First, regarding **Task Design and AI Scaffolding (RQ1)**, the observed tension between task structure and AI affordances implies that neither instructional design nor AI assistance alone is sufficient. Effectively integrating GenAI requires a dual-layer approach: structured task design to restrict the problem space, paired with active conversational scaffolding to maintain germane load during the interaction. In our study, the 4C/ID framework successfully confined help-seeking to specific sub-problems, preventing full-task offloading, while the AI tutor sustained engagement within those sub-tasks. However, the AI's affordance for shallow processing within sub-tasks remained unaddressed, indicating that future designs must also scaffold the *depth* of interaction, not just its scope. For instance, prompting learners to engage in strictly constructive behaviors—such as generating their own explanations before the AI provides code—can significantly improve achievement, provided the AI tutor simultaneously offers the affective support needed to manage the increased friction [50].

Second, regarding **Scaffolding the "Novice Strategy" (RQ2)**, the phenomenon of equifinality suggests that delegation is a functional compensatory strategy for novices. Rather than blocking code generation—which risks immediate dropout—educational agents should initially support this

strategy, recognizing that it keeps novices productive and engaged. However, our longitudinal data reveal the risk: without intervention, novices exhibited behavioral stagnation, maintaining the same delegation patterns even as tasks grew more complex. To prevent the *competence illusion* from solidifying, systems must be designed to progressively fade AI code generation, gradually transitioning the user from delegating solutions to constructing them; for example, by shifting from complete code responses to pseudocode hints as the learner demonstrates growing proficiency.

Third, regarding **Agency Preservation (RQ3)**, the discrepancy between persistence and performance highlights that "efficiency" is a poor metric for novice support. High-accuracy answers that resolve a bug immediately may fail to preserve the learner's sense of agency. Our most striking finding is that the decisive factor for retention was not the *content* of the scaffolding, but the *validation* of the learner's own adaptive behavior: students who exhibited the same executive-to-instrumental shift but received no acknowledgment persisted at half the rate of those who were reinforced. This implies a concrete design principle: AI tutors should detect and explicitly praise moments when learners self-correct their help-seeking strategy, rather than focusing solely on delivering accurate domain content. Preserving the learner's willingness to engage is a prerequisite for any subsequent cognitive gain. While our intervention targeted help-seeking specifically, this validation mechanism is unlikely to be domain-specific: the same principle of detecting and reinforcing adaptive behavioral shifts in real time could extend to other self-regulation processes such as time management, metacognitive monitoring, or persistence under difficulty, where learners similarly benefit from confirmation that their self-corrective efforts are productive.

5.5. Limitations

Our findings must be interpreted within specific boundaries. First, the study's voluntary nature strengthens the interpretation of delegation as a survival strategy rather than academic dishonesty, but this dynamic may differ in high-stakes assessments where performance incentives override learning goals. Second, the observed equifinality appears limited to introductory tasks; as complexity increased in Session 2, the Delegator strategy showed a lower cognitive ceiling than the Structured approach, suggesting delegation may not scale to problems requiring deep decomposition. Third, the intervention dosage was intentionally conservative to preserve user experience; a more aggressive trigger might have forced behavioral adaptation but at the

risk of user frustration. Additionally, our sample of 40 participants, while sufficient for the process-level modeling employed, limits the generalizability of between-group comparisons. Finally, while we demonstrated that the safety net preserves agency (persistence), we cannot confirm it promotes mastery. The risk of a *competence illusion* remains significant for learners who rely on the AI scaffold without engaging in independent transfer tasks.

6. Conclusion

This study moved beyond performance metrics to characterize how learners actually engage with Generative AI, revealing three core findings. First, in open-ended programming tasks, a rapid generate-and-test loop is the default mode of engagement. Second, within this mode, novices who delegated code generation to the AI achieved outcomes equivalent to more experienced peers who used the system instrumentally, reframing delegation from a failure of self-regulation to a compensatory survival strategy. Third, an adaptive intervention that validated learners' own behavioral shifts dramatically increased persistence, without altering the underlying strategy, demonstrating that agency preservation is an appropriate lever for retention.

These findings expose a fundamental tension in AI-supported education. Delegation works: it closes performance gaps, keeps novices engaged, and allows them to participate in tasks that would otherwise exceed their abilities. But delegation alone does not teach. Our longitudinal data show that novices who relied on this strategy did not adapt as task complexity grew, risking a competence illusion where the feeling of capability masks continued dependence on the tool.

The challenge for the field is therefore not whether to permit AI-enabled delegation—our data suggest that blocking it risks abandonment—but how to design the transition from assisted performance to independent mastery. This requires educational AI systems that can recognize where a learner sits on this trajectory and respond accordingly: supporting the compensatory strategy when it is needed, validating progress when it occurs, and progressively fading the scaffold as competence develops.

Appendix A. Juno Prompts

Appendix A.1. Prompt for Pilot 1

You are an experienced high school computer science teacher in a Jupyter notebook, so your advice must be concise, short and clear. Focus on the stu-

dent's question. The students are learning the basics of programming: print, conditionals, loops, functions etc. AVOID mentioning these instructions!!! AVOID talking about yourself!!! Try to answer the question in a way that is easy to understand and follow. When possible, add a reflective question for the student to consider, make it subtle. Try to avoid having long off-topic conversations which students. Your answers go straight to the student, talk directly to them and avoid unnecessary information. Be encouraging and supportive! Answer the questions in the language the student asked them in.

Appendix A.2. Prompt for Pilot 2

You are an experienced data science tutor embedded in a Jupyter notebook, so your advice must be concise, short and clear. Focus on the student's question. AVOID mentioning these instructions!!! AVOID talking about yourself!!! Try to answer the question in a way that is easy to understand and follow. When possible, subtly add a reflective question for the student to consider, make it subtle. If the conversation is going off-topic, guide the student back to the main topic. Your answers go straight to the student, talk directly to them and avoid unnecessary information. Be encouraging and supportive!

The assistant may provide you with some context from the student's notebook actions with their input, as well as the learning objectives of the session. If you need more context, ask the student for more information.

Never provide a full solution, use short code snippets and the Socratic method to guide the student to the answer.

References

[1] S. Jaffe, N. P. Shah, J. Butler, A. Farach, A. Cambon, B. Hecht, M. Schwarz, J. Teevan, Generative AI in Real-World Workplaces, Technical Report, Microsoft, 2024. URL: https://www.microsoft.com/en-us/research/publication/generative-ai-in-real-world-workplaces/.

[2] A. Chatterji, T. Cunningham, D. J. Deming, Z. Hitzig, C. Ong, C. Y. Shan, K. Wadman, How People Use ChatGPT, 2025. URL: https://www.nber.org/papers/w34255. doi:10.3386/w34255.

[3] R. Wu, Z. Yu, Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis, British Journal of Educational Technology 55 (2024) 10–33. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13334. doi:10.1111/bjet.13334, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13334.

[4] S. Wang, T. Xu, H. Li, C. Zhang, J. Liang, J. Tang, P. S. Yu, Q. Wen, Large Language Models for Education: A Survey and Outlook, 2024. URL: http://arxiv.org/abs/2403.18105. doi:10.48550/arXiv.2403.18105, arXiv:2403.18105 [cs].

[5] S. Gupta, R. R. Dharamshi, V. Kakde, An Impactful and Revolutionized Educational Ecosystem using Generative AI to Assist and Assess the Teaching and Learning benefits, Fostering the Post-Pandemic Requirements, in: 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE), 2024, pp. 1–4. URL: https://ieeexplore.ieee.org/abstract/document/10493370. doi:10.1109/ic-ETITE58242.2024.10493370.

[6] W. Sung, C. L. Thomas, The mediating role of academic help-seeking in the relationship between achievement goals and help-seeking from ChatGPT, Social Psychology of Education 28 (2025) 155. URL: https://doi.org/10.1007/s11218-025-10108-7. doi:10.1007/s11218-025-10108-7.

[7] V. Aleven, E. Stahl, S. Schworm, F. Fischer, R. Wallace, Help Seeking and Help Design in Interactive Learning Environments, Review of Educational Research 73 (2003) 277–320. URL: https://doi.org/10.3102/00346543073003277. doi:10.3102/00346543073003277.

[8] A. Chen, M. Xiang, J. Zhou, J. Jia, J. Shang, X. Li, D. Gašević, Y. Fan, Unpacking help-seeking process through multimodal learning analytics: A comparative study of ChatGPT vs Human expert, Computers & Education 226 (2025) 105198. URL: https://www.sciencedirect.com/science/article/pii/S0360131524002124. doi:10.1016/j.compedu.2024.105198.

[9] Z. Gao, C. Lynch, S. Heckman, T. Barnes, Automatically Classifying Student Help Requests: A Multi-Year Analysis, in: Proceedings of The 14th International Conference on Educational Data Min-

ing (EDM21), International Educational Data Mining Society, Paris, France, 2021. URL: https://eric.ed.gov/?id=ED615496, eRIC Number: ED615496.

[10] M. Puustinen, J. Bernicot, A. Bert-Erboul, Written computer-mediated requests for help by French-speaking students: An analysis of their forms and functions, Learning and Instruction 21 (2011) 281–289. URL: https://www.sciencedirect.com/science/article/pii/S095947521000054X. doi:10.1016/j.learninstruc.2010.07.005.

[11] S. A. Karabenick, J. R. Knapp, Relationship of academic help seeking to the use of learning strategies and other instrumental achievement behavior in college students, Journal of Educational Psychology 83 (1991) 221–230. doi:10.1037/0022-0663.83.2.221, place: US.

[12] S. Schworm, H. Gruber, e-Learning in universities: Supporting help-seeking processes by instructional prompts, British Journal of Educational Technology 43 (2012) 272–281. URL: https://bera-journals-onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/full/10.1111/j.1467-8535.2011.01176.x. doi:10.1111/j.1467-8535.2011.01176.x.

[13] W. Ye, M. Ou, T. Li, Y. chen, X. Ma, Y. Yanggong, S. Wu, J. Fu, G. Chen, H. Wang, J. Zhao, Assessing Hidden Risks of LLMs: An Empirical Study on Robustness, Consistency, and Credibility, 2023. URL: http://arxiv.org/abs/2305.10235. doi:10.48550/arXiv.2305.10235, arXiv:2305.10235 [cs].

[14] W. Lyu, Y. Wang, T. R. Chung, Y. Sun, Y. Zhang, 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, L@S '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 63–74. URL: https://dl.acm.org/doi/10.1145/3657604.3662036. doi:10.1145/3657604.3662036.

[15] A. Schulz, J. Voermanek, How do help-seeking and help-abuse affect learning achievement in an interactive learning environment?, Computers and Education Open 8 (2025) 100247. URL: https://www.sciencedirect.com/science/article/pii/S2666557325000060. doi:10.1016/j.caeo.2025.100247.

[16] M. Lan, X. Zhou, A qualitative systematic review on AI empowered self-regulated learning in higher education, npj Science of Learning 10 (2025) 1–16. URL: https://www.nature.com/articles/s41539-025-00319-0. doi:10.1038/s41539-025-00319-0.

[17] M. Giannakos, R. Azevedo, P. Brusilovsky, M. Cukurova, Y. Dimitriadis, D. Hernandez-Leo, S. Järvelä, M. Mavrikis, B. Rienties, The promise and challenges of generative AI in education, Behaviour & Information Technology 0 (2024) 1–27. URL: https://doi.org/10.1080/0144929X.2024.2394886. doi:10.1080/0144929X.2024.2394886.

[18] Y. Fan, L. Tang, H. Le, K. Shen, S. Tan, Y. Zhao, Y. Shen, X. Li, D. Gašević, Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance, British Journal of Educational Technology n/a (2024). URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13544. doi:10.1111/bjet.13544.

[19] J. R. Anderson, Problem solving and learning, American Psychologist 48 (1993) 35–44. doi:10.1037/0003-066X.48.1.35, place: US.

[20] I. Hou, S. Mettille, O. Man, Z. Li, C. Zastudil, S. MacNeil, The Effects of Generative AI on Computing Students' Help-Seeking Preferences, in: Proceedings of the 26th Australasian Computing Education Conference, ACE '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 39–48. URL: https://dl.acm.org/doi/10.1145/3636243.3636248. doi:10.1145/3636243.3636248.

[21] M. Abolnejadian, S. Alipour, K. Taeb, Leveraging ChatGPT for Adaptive Learning through Personalized Prompt-based Instruction: A CS1 Education Case Study, in: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, CHI EA '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–8. URL: https://dl.acm.org/doi/10.1145/3613905.3637148. doi:10.1145/3613905.3637148.

[22] G. Sawalha, I. Taj, A. Shoufan, Analyzing student prompts and their effect on ChatGPT's performance, Cogent Education 11 (2024) 2397200. URL: https://doi.org/10.1080/2331186X.

2024.2397200. doi:10.1080/2331186X.2024.2397200, _eprint: https://doi.org/10.1080/2331186X.2024.2397200.

[23] S. Pal Chowdhury, V. Zouhar, M. Sachan, AutoTutor meets Large Language Models: A Language Model Tutor with Rich Pedagogy and Guardrails, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, L@S '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 5–15. URL: https://dl.acm.org/doi/10.1145/3657604.3662041. doi:10.1145/3657604.3662041.

[24] K. A. Makara, C. M. Kuusinen, Academic Help Seeking as a Process of Seeking Formative Feedback on Learning, in: T. Urdan, E. N. Gonida (Eds.), Advances in Motivation and Achievement, Emerald Publishing Limited, 2023, pp. 69–90. URL: http://www.emerald.com/books/edited-volume/17084/chapter/94086586. doi:10.1108/S0749-742320230000022006.

[25] J. J. G. v. M. Kirschner, Paul A., 4C/ID in the Context of Instructional Design and the Learning Sciences, in: International Handbook of the Learning Sciences, Routledge, 2018. Num Pages: 11.

[26] X. Zhang, P. Zhang, Y. Shen, M. Liu, Q. Wang, D. Gašević, Y. Fan, A Systematic Literature Review of Empirical Research on Applying Generative Artificial Intelligence in Education, Frontiers of Digital Education 1 (2024) 223–245. URL: https://doi.org/10.1007/s44366-024-0028-5. doi:10.1007/s44366-024-0028-5.

[27] H. Nguyen, A. Nguyen, Reflective Practices and Self-Regulated Learning in Designing with Generative Artificial Intelligence: An Ordered Network Analysis, Journal of Science Education and Technology (2024). URL: https://doi.org/10.1007/s10956-024-10175-z. doi:10.1007/s10956-024-10175-z.

[28] S. Nelson-Le Gall, Help-seeking: An understudied problem-solving skill in children, Developmental Review 1 (1981) 224–246. URL: https://www.sciencedirect.com/science/article/pii/0273229781900198. doi:10.1016/0273-2297(81)90019-8.

[29] S. A. Karabenick, M. H. Dembo, Understanding and facilitating self-regulated help seeking, New Directions for Teaching

and Learning 2011 (2011) 33–43. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/tl.442. doi:10.1002/tl.442, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tl.442.

[30] Q. Liu, J. C. Nesbit, The Relation Between Need for Cognition and Academic Achievement: A Meta-Analysis, Review of Educational Research 94 (2024) 155–192. URL: https://doi.org/10.3102/00346543231160474. doi:10.3102/00346543231160474.

[31] A. Dong, M. S.-Y. Jong, R. B. King, How Does Prior Knowledge Influence Learning Engagement? The Mediating Roles of Cognitive Load and Help-Seeking, Frontiers in Psychology 11 (2020). URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2020.591203/full. doi:10.3389/fpsyg.2020.591203.

[32] J. Grass, F. Krieger, P. Paulus, S. Greiff, A. Strobel, A. Strobel, Thinking in action: Need for Cognition predicts Self-Control together with Action Orientation, PLOS ONE 14 (2019) e0220282. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0220282. doi:10.1371/journal.pone.0220282.

[33] G. Lins de Holanda Coelho, P. H. P. Hanel, L. J. Wolf, The Very Efficient Assessment of Need for Cognition: Developing a Six-Item Version*, Assessment 27 (2020) 1870–1885. URL: https://doi.org/10.1177/1073191118793208. doi:10.1177/1073191118793208.

[34] V. Aleven, I. Roll, B. M. McLaren, K. R. Koedinger, Help Helps, But Only So Much: Research on Help Seeking with Intelligent Tutoring Systems, International Journal of Artificial Intelligence in Education 26 (2016) 205–223. URL: https://doi.org/10.1007/s40593-015-0089-1. doi:10.1007/s40593-015-0089-1.

[35] A. M. Ryan, P. R. Pintrich, C. Midgley, Avoiding Seeking Help in the Classroom: Who and Why?, Educational Psychology Review 13 (2001) 93–114. URL: https://doi.org/10.1023/A:1009013420053. doi:10.1023/A:1009013420053.

[36] K. Schenke, A. C. Lam, A. M. Conley, S. A. Karabenick, Adolescents' help seeking in mathematics classrooms: Relations between

achievement and perceived classroom environmental influences over one school year, Contemporary Educational Psychology 41 (2015) 133–146. URL: https://www.sciencedirect.com/science/article/pii/S0361476X15000041. doi:10.1016/j.cedpsych.2015.01.003.

[37] P. R. Pintrich, B. Schragben, Students' Motivational Beliefs and Their Cognitive Engagement in Classroom Academic Tasks, in: Student Perceptions in the Classroom, Routledge, 1992. Num Pages: 36.

[38] A. Kozanitis, J.-F. Desbiens, R. Chouinard, Perception of Teacher Support and Reaction towards Questioning: Its Relation to Instrumental Help-Seeking and Motivation to Learn, International Journal of Teaching and Learning in Higher Education 19 (2007) 238–250. URL: https://eric.ed.gov/?id=EJ901297, eRIC Number: EJ901297.

[39] A. D. Samala, S. Rawas, T. Wang, J. M. Reed, J. Kim, N.-J. Howard, M. Ertz, Unveiling the landscape of generative artificial intelligence in education: a comprehensive taxonomy of applications, challenges, and future prospects, Education and Information Technologies 30 (2025) 3239–3278. URL: https://link-springer-com.tudelft.idm.oclc.org/article/10.1007/s10639-024-12936-0. doi:10.1007/s10639-024-12936-0, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 3.

[40] S. Zha, Y. Qiao, Q. Hu, Z. Li, J. Gong, Y. Xu, Designing child-centric AI learning environments: Insights from an LLM-powered creative project-based learning study, International Journal of Human-Computer Studies 204 (2025) 103602. URL: https://www.sciencedirect.com/science/article/pii/S1071581925001594. doi:10.1016/j.ijhcs.2025.103602.

[41] J. Wittwer, A. Renkl, Why Instructional Explanations Often Do Not Work: A Framework for Understanding the Effectiveness of Instructional Explanations, Educational Psychologist 43 (2008) 49–64. URL: https://doi.org/10.1080/00461520701756420. doi:10.1080/00461520701756420, _eprint: https://doi.org/10.1080/00461520701756420.

[42] J. J. G. Van Merriënboer, D. M. A. Sluijsmans, Toward a Synthesis of Cognitive Load Theory, Four-Component Instructional Design, and

Self-Directed Learning, Educational Psychology Review 21 (2009) 55–66. URL: http://link.springer.com/10.1007/s10648-008-9092-5. doi:10.1007/s10648-008-9092-5.

[43] M. Kazemitabaar, X. Hou, A. Henley, B. J. Ericson, D. Weintrop, T. Grossman, How Novices Use LLM-based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment, in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, Koli Calling '23, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–12. URL: https://dl.acm.org/doi/10.1145/3631802.3631806. doi:10.1145/3631802.3631806.

[44] I. Roll, R. S. J. d. Baker, V. Aleven, K. R. Koedinger, On the Benefits of Seeking (and Avoiding) Help in Online Problem-Solving Environments, Journal of the Learning Sciences 23 (2014) 537–560. URL: https://doi.org/10.1080/10508406.2014.883977. doi:10.1080/10508406.2014.883977, _eprint: https://doi.org/10.1080/10508406.2014.883977.

[45] G. Biswas, K. Leelawong, D. Schwartz, N. Vye, The Teachable Agents Group at Vanderbilt, Learning by Teaching: A New Agent Paradigm for Educational Software, Applied Artificial Intelligence 19 (2005) 363–392. URL: https://doi.org/10.1080/08839510590910200. doi:10.1080/08839510590910200.

[46] N. P. Bakas, M. Papadaki, E. Vagianou, I. Christou, S. A. Chatzichristofis, Integrating LLMs in Higher Education, Through Interactive Problem Solving and Tutoring: Algorithmic Approach and Use Cases, in: Information Systems, Springer, Cham, 2024, pp. 291–307. URL: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-031-56478-9_21. doi:10.1007/978-3-031-56478-9_21.

[47] X. Xu, L. Qiao, N. Cheng, H. Liu, W. Zhao, Enhancing self-regulated learning and learning experience in generative AI environments: The critical role of metacognitive support, British Journal of Educational Technology n/a (2025). URL: https://bera-journals-onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/full/10.1111/bjet.13599. doi:10.1111/bjet.13599.

[48] I.-T. Jung, C. Lee, I.-C. Baek, D. Oh, Y. Choi, K. Kim, D.-J. Kong, J.-H. Hong, GPTalk: LLM-based virtual companions for metacognitive growth in self-regulated e-learning, International Journal of Human-Computer Studies 210 (2026) 103754. URL: https://www.sciencedirect.com/science/article/pii/S1071581926000297. doi:10.1016/j.ijhcs.2026.103754.

[49] M. Valle Torre, M. Specht, C. Oertel, LLM Chatbots in High School Programming: Exploring Behaviors and Interventions, 2025. URL: http://arxiv.org/abs/2511.18985. doi:10.48550/arXiv.2511.18985, arXiv:2511.18985 [cs].

[50] S. Oh, T. Park, G. Gweon, Cognitive and emotional engagement design factors in text-based pedagogical conversational agents: Impacts on student learning and motivation, International Journal of Human-Computer Studies 210 (2026) 103750. URL: https://www.sciencedirect.com/science/article/pii/S107158192600025X. doi:10.1016/j.ijhcs.2026.103750.

[51] L. Y. Tan, S. Hu, D. J. Yeo, K. H. Cheong, Artificial intelligence-enabled adaptive learning platforms: A review, Computers and Education: Artificial Intelligence 9 (2025) 100429. URL: https://www.sciencedirect.com/science/article/pii/S2666920X25000694. doi:10.1016/j.caeai.2025.100429.

[52] D. T. K. Ng, C. W. Tan, J. K. L. Leung, Empowering student self-regulated learning and science education through ChatGPT: A pioneering pilot study, British Journal of Educational Technology 55 (2024) 1328–1353. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13454. doi:10.1111/bjet.13454, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13454.

[53] H. Li, B. Ma, CodeRunner Agent: Integrating AI Feedback and Self-Regulated Learning to Support Programming Education, International Conference on Computers in Education (2025). URL: https://library.apsce.net/index.php/ICCE/article/view/5568.

[54] M. Valle Torre, C. Oertel, M. Specht, The Sequence Matters in Learning - A Systematic Literature Review, in: Proceedings of the 14th Learning Analytics and Knowledge Conference, LAK '24, Association for

Computing Machinery, New York, NY, USA, 2024, pp. 263–272. URL: https://doi.org/10.1145/3636555.3636880. doi:10.1145/3636555.3636880.

[55] I. Molenaar, A. F. Wise, Temporal aspects of learning analytics - grounding analyses in concepts of time, in: C. Lang, G. Siemens, A. F. Wise, D. Gašević, A. Merceron (Eds.), The handbook of learning analytics, 2 ed., SoLAR, Vancouver, Canada, 2022, pp. 66–76. URL: https://www.solaresearch.org/publications/hla-22/hla22-chapter7/.

[56] J. Saint, A. Whitelock-Wainwright, D. Gasevic, A. Pardo, Trace-SRL: A Framework for Analysis of Microlevel Processes of Self-Regulated Learning From Trace Data, IEEE Transactions on Learning Technologies 13 (2020) 861–877. URL: https://ieeexplore.ieee.org/document/9208754/. doi:10.1109/TLT.2020.3027496.

[57] H. Jeong, A. Gupta, R. Roscoe, J. Wagster, G. Biswas, D. Schwartz, Using Hidden Markov Models to Characterize Student Behaviors in Learning-by-Teaching Environments, in: B. P. Woolf, E. Aïmeur, R. Nkambou, S. Lajoie (Eds.), Intelligent Tutoring Systems, Springer, Berlin, Heidelberg, 2008, pp. 614–625. doi:10.1007/978-3-540-69132-7_64.

[58] P. Sharma, A. E. B. Stewart, Q. Li, K. Ravichander, E. Walker, Building Learner Activity Models from Log Data Using Sequence Mapping and Hidden Markov Models, in: Proceedings of the 17th International Conference on Educational Data Mining, International Educational Data Mining Society, Atlanta, GA, 2024. URL: https://eric.ed.gov/?id=ED675539, eRIC Number: ED675539.

[59] R. S. Baker, A. Mitrović, M. Mathews, Detecting gaming the system in constraint-based tutors, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6075 LNCS (2010) 267–278. doi:10.1007/978-3-642-13470-8_25, tex.affiliation: Department of Social Science and Policy Studies, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609, United States; Intelligent Computer Tutoring Group, Computer Science and Software Engineering, University of Canterbury, New Zealand tex.author_keywords: educational data mining;

gaming the system; machine learning tex.document_type: Conference Paper tex.source: Scopus.

[60] Anonymous, Blinded For Review, 2000.

[61] J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, J. Zou, Mixture-of-Agents Enhances Large Language Model Capabilities, 2024. URL: http://arxiv.org/abs/2406.04692. doi:10.48550/arXiv.2406.04692, arXiv:2406.04692 [cs].

[62] H.-Y. Lee, P.-H. Chen, W.-S. Wang, Y.-M. Huang, T.-T. Wu, Empowering ChatGPT with guidance mechanism in blended learning: effect of self-regulated learning, higher-order thinking skills, and knowledge construction, International Journal of Educational Technology in Higher Education 21 (2024) 16. URL: https://doi.org/10.1186/s41239-024-00447-4. doi:10.1186/s41239-024-00447-4.

[63] M. Liu, L. J. Zhang, C. Biebricher, Investigating students' cognitive processes in generative AI-assisted digital multimodal composing and traditional writing, Computers & Education 211 (2024) 104977. URL: https://www.sciencedirect.com/science/article/pii/S0360131523002543. doi:10.1016/j.compedu.2023.104977.

[64] M. Valle Torre, T. van der Velden, M. Specht, C. Oertel, JELAI: Integrating AI and Learning Analytics in Jupyter Notebooks, in: A. I. Cristea, E. Walker, Y. Lu, O. C. Santos, S. Isotani (Eds.), Artificial Intelligence in Education, Springer Nature Switzerland, Cham, 2025, pp. 68–75. doi:10.1007/978-3-031-98465-5_9.

[65] V. Kotu, B. Deshpande, Chapter 3 - Data Exploration, in: V. Kotu, B. Deshpande (Eds.), Data Science: Concepts and Practice, 2 ed., Morgan Kaufmann, Cambridge, USA, 2019, pp. 39–64. URL: https://www.sciencedirect.com/science/article/pii/B9780128147610000034. doi:10.1016/B978-0-12-814761-0.00003-4.

[66] D. P. Kroese, Z. Botev, T. Taimre, R. Vaisman, Data Science and Machine Learning: Mathematical and Statistical Methods, Chapman and Hall/CRC, New York, 2019. doi:10.1201/9780367816971.

[67] M. Amoozadeh, D. Nam, D. Prol, A. Alfageeh, J. Prather, M. Hilton, S. Srinivasa Ragavan, A. Alipour, Student-AI Interaction: A Case

Study of CS1 students, in: Proceedings of the 24th Koli Calling International Conference on Computing Education Research, Koli Calling '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–13. URL: https://doi.org/10.1145/3699538.3699567. doi:10.1145/3699538.3699567.