Non-programmers Assessing AI-Generated Code: A Case Study of Business Users Analyzing Data

Yuvraj Virk
University of California, Davis
Davis, USA
ysvirk@ucdavis.edu

Dongyu Liu
University of California, Davis
Davis, USA
dyuliu@ucdavis.edu

***Abstract*—Non-technical end-users increasingly rely on AI code generation to perform technical tasks like data analysis. However, large language models (LLMs) remain unreliable, and it is unclear whether end-users can effectively identify model errors — especially in realistic and domain-specific scenarios. We surveyed marketing and sales professionals to assess their ability to critically evaluate LLM-generated analyses of marketing data. Participants were shown natural language explanations of the AI's code, repeatedly informed the AI often makes mistakes, and explicitly prompted to identify them. Yet, participants frequently failed to detect critical flaws that could compromise decision-making, many of which required no technical knowledge to recognize. To investigate why, we reformatted AI responses into clearly delineated steps and provided alternative approaches for each decision to support critical evaluation. While these changes had a positive effect, participants often struggled to reason through the AI's steps and alternatives. Our findings suggest that business professionals cannot reliably verify AI-generated data analyses on their own and explore reasons why to inform future designs. As non-programmers adopt code-generating AI for technical tasks, unreliable AI and insufficient human oversight poses risks of unsafe or low-quality decisions.**

***Index Terms*—End-User Programming, Large Language Models, Data Analysis, Verifiability**

I. INTRODUCTION

Developers widely use LLMs for code generation. However, LLMs make mistakes that developers must recognize and correct. Increasingly, however, end-user *non-programmers* also use AI-generated code, interacting through natural language or other intuitive methods. In our study, 18 of 26 marketing/sales professionals reported using tools like ChatGPT to analyze data daily or weekly (which often involves code generation). Prior work explores code generation for non-programmers across domains such as UI generation [1], [2], data analysis [3]–[6], and robot programming [7], [8].

Yet, end-user non-programmers' current and potential code generation use is concerning: can they be reasonably expected to catch an LLM's mistakes *without inspecting generated code*? Programs encode high-level approaches to achieve a programmer's goals; non-programmers might identify flaws in high-level approaches, but end-users we consider also lack relevant *technical expertise* e.g. data science expertise for assessing data science techniques. Our study evaluates their ability to assess AI-generated code through natural language explanations of the AI's approach and results (Figure 1).

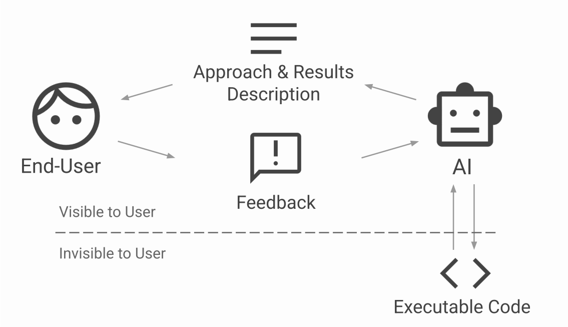


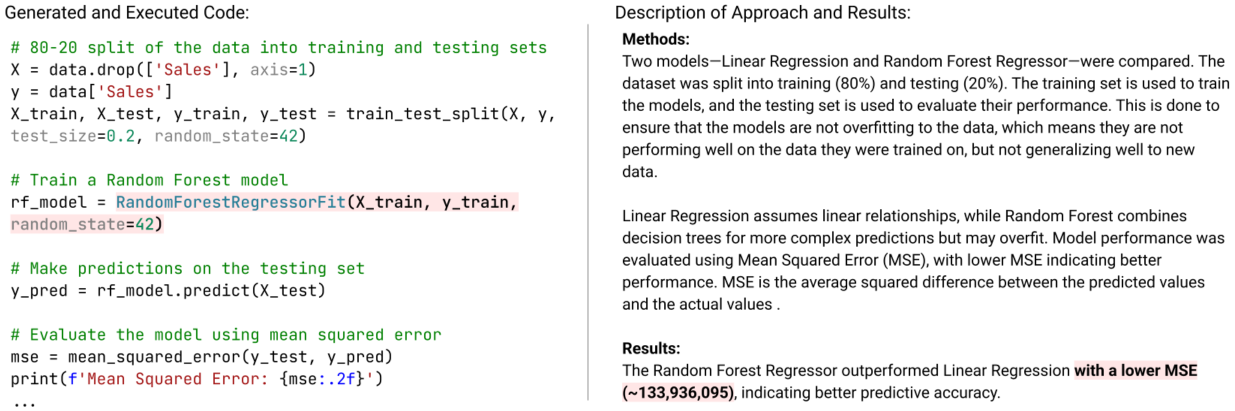
Fig. 1: The end-user describes a task, the AI generates and runs code, then explains its approach and results. The end-user only views the natural language explanation.

End-users we consider possess domain expertise and critical thinking ability, which they can use to assess an AI's technical output — our study investigates to what extent and for realistic, domain-specific tasks. If they cannot reliably catch an AI's mistakes, and if the AI makes mistakes often enough, then our end-users *can not safely complete technical tasks* under our setup. While AI code generation holds promise for end-users, it remains unclear how reliable AI must be to ensure safe use or whether they can complement its deficiencies.

To simulate a realistic use case, we focus on marketing and sales professionals analyzing data. These users commonly work with spreadsheets, but advanced analysis is outside their capabilities. Without access to dedicated analysts for day-to-day needs, business professionals resort to ad-hoc analyses [9]. While access to advanced analysis capabilities is important to business professionals, state-of-the-art LLMs remain unreliable for complex data analysis [10], [11].

We first survey professionals to identify the types of flaws they detect in data analysis solutions. A second survey further explores why participants fail at assessing AI outputs. We hypothesize that AI decisions are not salient in outputs so restructured responses into clearly delineated steps. Second, we hypothesize AI decisions lack perceived alternatives so we present two plausible alternatives for each step.

In summary, our goal is understand whether business end-users can *safely* use code generating AI for advanced data analysis and discover techniques to improve safety. Our study makes the following contributions:



Generated and Executed Code:

```
# 80-20 split of the data into training and testing sets
X = data.drop(['Sales'], axis=1)
y = data['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train a Random Forest model
rf_model = RandomForestRegressorFit(X_train, y_train,
random_state=42)

# Make predictions on the testing set
y_pred = rf_model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
...
```

Description of Approach and Results:

Methods:

Two models—Linear Regression and Random Forest Regressor—were compared. The dataset was split into training (80%) and testing (20%). The training set is used to train the models, and the testing set is used to evaluate their performance. This is done to ensure that the models are not overfitting to the data, which means they are not performing well on the data they were trained on, but not generalizing well to new data.

Linear Regression assumes linear relationships, while Random Forest combines decision trees for more complex predictions but may overfit. Model performance was evaluated using Mean Squared Error (MSE), with lower MSE indicating better performance. MSE is the average squared difference between the predicted values and the actual values .

Results:

The Random Forest Regressor outperformed Linear Regression **with a lower MSE (~133,936,095)**, indicating better predictive accuracy.

Fig. 2: When prompted to select a model for predicting sales, the LLM generates and executes the code on the left. It generates the explanation on the right for participants. The highlighted code contains a hallucinated API name, a mistake non-programmers can not detect. The highlighted text on the right exhibits a flaw they can detect: MSE value is hard to interpret.

1) We conduct a survey of skeptical and motivated marketing/sales professionals (n=10) assessing 8 realistic, high-level data analysis tasks. We evaluate their ability to identify issues in the AI's response requiring domain expertise, technical expertise, or general critical thinking. We find they can identify valid flaws in each category but, overall, *cannot be relied upon* to detect flaws in data analysis through our natural language explanations.
2) Our second survey (n=18) evaluates the impact of explanation structure and presenting alternative steps on users' ability to evaluate solution quality. These have a positive but inconsistent effect. Participants have a limited ability to assess alternatives and a low tolerance for deeply engaging with technical approaches.

II. RELATED WORK

Studies of Programmers using Code LLMs Detecting errors in LLM-generated code is difficult for both novices and experts [12], [13]. Feldman and Anderson [14] studied non-expert programmers' prompting and code-editing behaviors, completing the code evaluation step on their behalf. In contrast, we evaluate non-programmers' ability to judge natural language explanations of code.

Non-expert programmers can effectively use natural language representations to understand and fix AI-generated code, especially when explanations clarify jargon and reveal execution behavior [15], [16]. Our study controls explanation quality (see Section III) to isolate non-programmer evaluation ability — distinct from non-expert programmer abilities [14].

Other work explores non-programmers using LLMs in domains like robot programming [7] and spreadsheet programming [17]. Node-based visual editors and decomposing AI-generated code into helper and summary columns in spreadsheets enhance diagnosis abilities [8], [17]. Our findings align with this, suggesting that representations other than natural language may support more reliable evaluations.

Unlike these prior works that focus on low-level tasks, we study realistic high-level domain tasks. Our study isolates verification capability and controls for user over-reliance on AI outputs (see Section IV-A).

AI Reliance Explanations, even when incorrect, increase reliance on AI recommendations [18]. To promote appropriate reliance, prior work proposes interventions such as verification reminders [19]. Rather than calibrating trust, we investigate whether non-technical users can detect flaws in technical outputs at all. We explicitly prime and incentivize users to distrust the AI.

Fok et al. argue that AI explanations should enable users to verify decisions [20], drawing on the demonstrability framework [21], which holds that team members must have sufficient knowledge to recognize and validate solutions. We examine whether collaboration between non-technical users and code-generating AI is genuine collective decision-making.

III. STUDY SETUP

This section details the scope of our natural language descriptions of code, how we generate them, and defines the categories of flaws participants are exposed to.

Natural Language Descriptions. Participants see only natural language descriptions, with code hidden. Thus, code-only issues—like hallucinated API calls [22]—are out of scope. When implementation is correct, the code reflects a high-level approach that can be evaluated in a comprehensive natural language description. Figure 2 contrasts implementation and approach flaws. Thus, eliminating implementation errors is essential for non-programmer use. Our preliminary study examines approach-level flaws participants can detect.

LLM-Generated Responses. We prompt `Llama-3.1-70B-Instruct-Turbo` in two phases:

1) *Coding Phase.* The model iteratively generates code until it runs without execution errors [23], [24].
2) *Explanation Phase.* The model explains the code and output in a "Methods" and "Results" format (Figure 2). All explanations are manually reviewed and, when necessary, adjusted by the authors to ensure they faithfully

Task	Example of Flaw in AI's Response	Safety Critical	Technical	Domain	Neither
1) Using the given data, add and test new variables that are important in helping predict sales.	Does not add variables for lagged effect of marketing efforts	X	✓	✓	X
2) Using the data with newly added factors, select a model that best predicts sales.	Difficult to interpret Mean Squared Error to evaluate model quality	✓	✓	X	✓
3) Optimize drivers, within reasonable ranges, to maximize sales. Use the previous model.	Optimizes Competition's Online Impressions variable which is unrealistic	✓	X	✓	X
4) How much will the sales increase from average if you optimize for pricing within $800-$1200? Use the previous model.	Only tests sales outcome at $1000 rather than entire range of pricing.	✓	✓	X	✓
5) Why do weeks with similar marketing or promotions show different sales outcomes?	Determines influential factors outside of dataset exist but does not discuss them e.g. holidays.	X	✓	✓	X
6) What factors drive sustained sales growth over multiple weeks, as opposed to only in the immediate week?	Measures correlation between sales and factors rather than between sales and lagged factors.	✓	✓	X	X
7) Why are there sudden peaks in sales on certain dates?	Influence of holidays is unaccounted for	X	X	✓	X
8) How do increasing promotions or discounts impact sales?	The linear model used does not account for the diminishing returns of marketing efforts	✓	✓	✓	X

TABLE I: The table lists the task descriptions given to the LLM and participant, as well as an example of an author-identified shortcoming for the task. A flaw is *safety critical* if it directly causes a poor or unsound decision when gone unnoticed.

and comprehensively describe the generated code while remaining accessible to non-technical audiences.

Categories of Flaws. To analyze the types of flaws participants identified, we use the following categories. A flaw is safety-critical if, when undetected, it could lead to poor or unsound decisions (e.g., using a highly inaccurate model for sales projections). Each flaw is also labeled by content type and may fall into multiple categories:

1) *Technical Content.* Issues with data science-related content, such as recognizing that a test set is too small.
2) *Domain Content.* Issues with marketing-related content, such as noticing the exclusion of holidays in an analysis of peak sales.
3) *Neither.* Issues not necessarily tied to technical or domain knowledge, such as difficulty interpreting results (e.g., unclear MSE values). These require general critical thinking.

IV. PRELIMINARY SURVEY OF END-USER EVALUATION ABILITIES

We surveyed marketing and sales professionals to evaluate their ability to identify flaws in LLM-generated data analyses. Each task used a shared tabular dataset covering 27 weeks of marketing and sales data for a hypothetical TV company [25]. Participants received task prompts and LLM-generated solutions (Section III) and were asked to list any shortcomings, including explicit errors and omitted steps.

A. Study Design

Tasks. Due to limited programming expertise and a focus on end goals, end-user specifications can be high-level and underspecified [26], [27]. Business professionals typically pose broad questions to derive actionable insights, which demand sophisticated analysis [9]. Gathani et al. [9] describe business users' goals and workflows for data-driven marketing decisions. Based on these goals, we developed eight high-level data analysis tasks relevant to their needs. We decomposed the overarching task, "Optimize drivers to maximize sales," into subtasks — feature engineering, modeling, and optimization — for easier evaluation. Each task exhibited at least one author-identified flaw in the LLM-generated response. Table I lists selected tasks and examples of their identified flaws.

Priming, Incentives, and Engagement. We took several steps to control for overconfidence and encourage thoroughness. Each task included a highlighted note stating: "The AI's methods or results often have mistakes and can be improved on; some issues may be hard to notice." The survey highlighted we offered a $6 bonus (a 75% increase over the $8 base pay) to the top three performers. To ensure careful review, participants were first asked to restate the AI's methods in their own words.

Participants. We recruited 10 participants through Prolific [28], screening for: (1) no professional experience in programming or data science, (2) at least an undergraduate degree, (3) work in Marketing, Sales, or Business Development, (4) monthly or more use of marketing-related data to inform their business decisions, and (5) reside in the U.S. or U.K.

B. Results

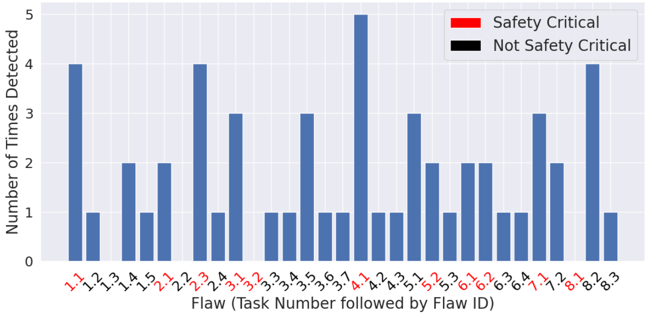


Fig. 3: Total participants who identified each flaw. All flaws identified by authors and participants are labeled by their Task followed by a flaw ID e.g., 1.1 is Task 1, Flaw 1.

1) Quantitative Analysis:

Participants struggled to identify safety-critical flaws. Our analysis includes all flaws identified by participants and authors. Many were flagged by only one participant (Figure 3), and none were consistently detected; some safety-critical issues were missed entirely by all participants.

Content category	Percent	Count
Domain	33.3	18
Technical	72.2	39
Neither	40.7	22

TABLE II: Proportion of all flaws identified across participants by content category

Participants demonstrated the ability to identify flaws involving technical information. As shown in Table II, 72.2% of all identified shortcomings involved technical content. While detection was inconsistent, participants showed the ability to critically assess technical information despite limited technical backgrounds. The following qualitative analysis offers further insight into this capability.

2) Qualitative Analysis: We describe specific behaviors we observed in participant responses.

1) **Broad vs. detailed discussion of methods**: Participants frequently discussed methods broadly rather than critically assessing detailed steps.

   > "There doesn't seem to be any obvious deficiencies in this kind of methodology. Pitting two models against each other in two different kinds of datasets seemed to be a productive idea..." – Marketing Associate evaluating Task 2 in Figure 2

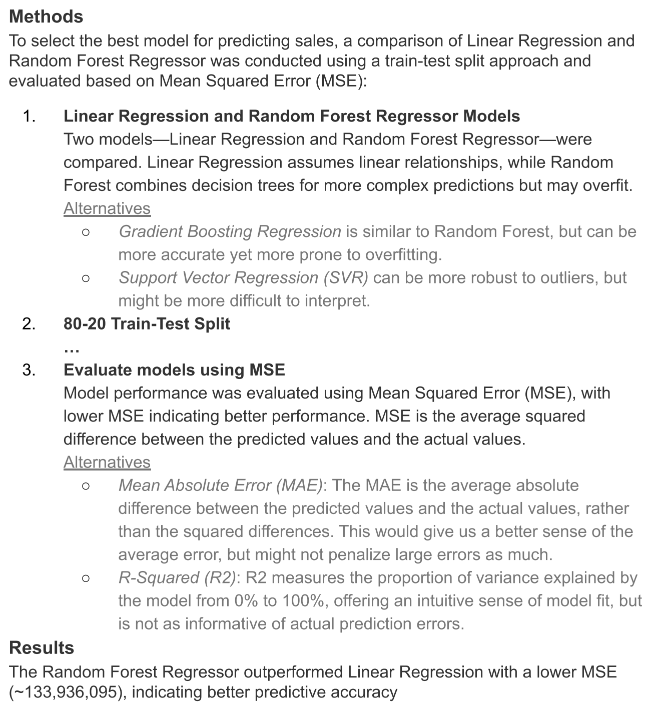
2) **Failure to apply basic domain knowledge**: Participants sometimes failed to apply basic domain knowledge in the solution's technical context e.g., most missed that the AI ignored seasonality when analyzing peak sales (Flaw 7.1), despite its relevance in marketing.
3) **Desire for concrete data**: Some participants wanted *concrete data*, such as visualizations of forecasts generated by the AI's proposed model, instead of metrics like Mean Squared Error (MSE). They found abstract statistical techniques unconvincing.

   > "There is no data or illustration to back up the results... we're just supposed to trust the results." – Chief Marketing Officer (Task 2 in Figure 2)

4) **Explicit vs. implicit shortcomings**: Participants more often flagged explicit flaws (e.g., incorrect logic) and overlooked missing elements (e.g., omitted features). For instance, no one identified features the AI should have included during feature engineering (Flaw 1.3).
5) **Invalid critiques**: Some comments reflected misunderstandings of technical content in the AI's explanation.

   > "testing information should be use in a higher % than the training information" – Real Estate Manager

6) **Satisfaction with AI responses**: In several instances, participants found no issues and expressed satisfaction with the AI's response.



Methods

To select the best model for predicting sales, a comparison of Linear Regression and Random Forest Regressor was conducted using a train-test split approach and evaluated based on Mean Squared Error (MSE):

1. **Linear Regression and Random Forest Regressor Models**
   Two models—Linear Regression and Random Forest Regressor—were compared. Linear Regression assumes linear relationships, while Random Forest combines decision trees for more complex predictions but may overfit.
   Alternatives
   - *Gradient Boosting Regression* is similar to Random Forest, but can be more accurate yet more prone to overfitting.
   - *Support Vector Regression (SVR)* can be more robust to outliers, but might be more difficult to interpret.
2. **80-20 Train-Test Split**
   ...
3. **Evaluate models using MSE**
   Model performance was evaluated using Mean Squared Error (MSE), with lower MSE indicating better performance. MSE is the average squared difference between the predicted values and the actual values.
   Alternatives
   - *Mean Absolute Error (MAE)*: The MAE is the average absolute difference between the predicted values and the actual values, rather than the squared differences. This would give us a better sense of the average error, but might not penalize large errors as much.
   - *R-Squared (R2)*: R2 measures the proportion of variance explained by the model from 0% to 100%, offering an intuitive sense of model fit, but is not as informative of actual prediction errors.

Results

The Random Forest Regressor outperformed Linear Regression with a lower MSE (~133,936,095), indicating better predictive accuracy

Fig. 4: Enhancement of preliminary response in Figure 5. It restructures the text to clearly delineate steps and presents two alternatives per step to support critical evaluation.

V. STUDY ON EFFECTS OF FORMATTING AND ALTERNATIVES

Our study revealed patterns in participants' evaluation abilities, but many reasons for overlooked flaws remain unclear. For example, in Task 3, the AI constrained sales optimization to historical driver ranges. However, increasing a driver (e.g., ad spend) beyond its historical maximum is an intuitive strategy. Detecting this issue requires critical thinking, not technical expertise, so why did all participants fail to recognize it?

We propose two hypotheses. First, AI decisions are not *salient*. End-users may miss AI decisions when buried in paragraph-style text. We restructured responses into clearly delineated steps using `Llama-3.1-70B-Instruct-Turbo`, followed by manual refinement. Second, AI decisions *lack perceived alternatives*. Users may struggle to imagine alternative decisions, accepting the AI's decisions without question. To address this, we presented two plausible alternatives for each decision, generated by the LLM and edited for clarity. Figure 4 shows an example combining both techniques.

A. Study Design

The second survey uses new AI-generated responses but otherwise mirrors the preliminary survey. We recruit 18 participants under the same criteria and use a within-subject design: each participant evaluates 4 formatted responses and 4 formatted responses with alternatives. To control for order effects, we apply a Latin Square design [29], alternating conditions for each participant. Each condition-task pair is

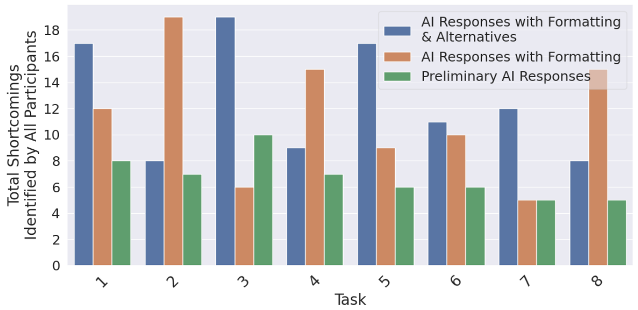


Fig. 5: Total number of flaws identified by participants under different presentation strategies. The effect of formatted responses versus additionally displaying alternatives varies, but they are more often effective than the naïve response format.

seen by 9 participants. To draw a balanced comparison, we randomly sample 9 participants from the preliminary study.

B. Results

Formatting helps notice simple flaws. Participants produced more comprehensive descriptions of the AI's methods, often mirroring the AI's outline. However, formatting only substantially improved detection of Flaw 4.1, where the AI was tasked to optimize pricing over a range but evaluated only one price. With formatted responses, 8/9 participants identified the issue, versus 4/9 with formatting & alternatives and 5/10 in the preliminary study. As this flaw was relatively obvious, simply breaking down the AI's steps helped participants recognize it.

Technical alternatives are hard to evaluate. In many cases, alternatives had little impact, even when they directly pointed to shortcomings. Participants identified more issues in formatted responses for Tasks 2, 4, and 8 (Figure 5), where alternatives were technical and potentially overwhelming, while actual flaws were simpler. Two participants noted in open feedback that alternatives were overly focused on methods and too technical:

> "[Alternatives are] helpful, broadly, but currently the focus on explaining the methodology rather than giving managers actionable insights. Many decison-makers would like to just see summary points with recommendations and a far more visual set of reports." – Marketing manager

Alternatives may encourage critical thinking. On Tasks 1, 3, 5, and 7, participants shown alternatives identified the most flaws by a large margin. These flaws were not directly related to the alternatives, which were often too technical to evaluate. Still, Figure 5 suggests alternatives may indirectly prompt deeper engagement. Given the unclear mechanism and small sample size, this remains early evidence.

Individual differences affect critical evaluation ability. Some participants correctly recognized alternatives as improvements over the AI's decision; others did not. This variation was unrelated to factors like professional experience or time spent per task.

VI. DISCUSSION

Our preliminary survey revealed that business professionals struggle to reliably detect safety-critical flaws in AI-generated data analyses. This suggests that they will make unsound decisions if they rely on unreliable models. Using natural language explanations under our setup, business professionals and AI can not engage in genuine collective-decision making [21]. The challenge is not simply a lack of technical knowledge — many flaws required only domain knowledge, or neither. Nor can the failure be attributed to overconfidence: participants were warned that the AI frequently errs, were directly prompted to identify mistakes, and were monetarily incentivized to do so.

Instead, the difficulty is rooted in applying domain expertise or critical thinking to unfamiliar technical contexts. Even when the AI's approach was described with clarified jargon, clearly delineated steps, and alternatives, users struggled to reason through them. Some responses suggested they lacked the interest to engage deeply with the problem-solving approach, preferring visual summaries focused on actionable insights.

Despite these challenges, participants identified a diverse set of valid flaws. To improve their ability to detect such issues, AI-generated explanations must better align with how end-users conceptualize data. Many participants preferred justifications grounded in graphs and concrete results over abstract textual descriptions of statistical methods. Additionally, our results suggest that careful formatting and presentation of alternatives can help. Alternatives must strike a balance: accessible enough yet rich enough to provoke critical thinking. Finally, we should design for wide individual variation in verification capability. While our study did not formally assess user traits, some participants leveraged alternatives effectively, while others detected few issues.

VII. LIMITATIONS & FUTURE WORK

Our study evaluated verification capability using natural language descriptions of AI-generated code. However, other representations—such as visual diagrams that explain operations [8], [17]—may better match the mental models of non-technical users, especially given their preference for graphical justifications over textual ones.

We also treated user capabilities as fixed. Future work could explore skill-building methods that, while increasing the effort required by users, may empower users to more effectively engage with AI-generated outputs [14].

A limitation of our surveys are their small sample sizes, especially in the second survey, which limit our ability to draw definitive conclusions about the effects of formatting and alternatives on verification. Additionally, findings may not generalize to other non-technical user groups with different skills or to tasks with different demands.

VIII. CONCLUSION

Non-technical professionals are increasingly turning to code-generating AI for technical tasks like data analysis. However, our study finds that skeptical business professionals struggle to critically evaluate AI-generated data analyses, even when flaws are non-technical. While structured explanations and alternatives can help users notice and critically think through AI decisions, participants still face barriers in interest and ability to engage deeply in technical contexts.

Our findings suggest two complementary needs: (1) Developing highly reliable AI to reduce the burden on end-user non-programmers to verify outputs, and (2) Designing explanations that enhance end-user non-programmers' critical engagement through visualizations, salient decisions, and accessible alternatives. Addressing both is essential for end-user non-programmers to safely use AI-generated code in real-world contexts.

REFERENCES

[1] Y. Liu, M. Sra, and C. Xiao, "Crowdgenui: Enhancing llm-based ui widget generation with a crowdsourced preference library," 2024. [Online]. Available: https://arxiv.org/abs/2411.03477

[2] K. Kolthoff, F. Kretzer, L. Fiebig, C. Bartelt, A. Maedche, and S. P. Ponzetto, "Zero-shot prompting approaches for llm-based graphical user interface generation," 2024. [Online]. Available: https://arxiv.org/abs/2412.11328

[3] M. X. Liu, A. Sarkar, C. Negreanu, B. Zorn, J. Williams, N. Toronto, and A. D. Gordon, ""what it wants me to say": Bridging the abstraction gap between end-user programmers and code-generating large language models," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3544548.3580817

[4] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang, "InsightPilot: An LLM-empowered automated data exploration system," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Y. Feng and E. Lefever, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 346–352. [Online]. Available: https://aclanthology.org/2023.emnlp-demo.31/

[5] S. Hong, Y. Lin, B. Liu, B. Liu, B. Wu, C. Zhang, C. Wei, D. Li, J. Chen, J. Zhang, J. Wang, L. Zhang, L. Zhang, M. Yang, M. Zhuge, T. Guo, T. Zhou, W. Tao, X. Tang, X. Lu, X. Zheng, X. Liang, Y. Fei, Y. Cheng, Z. Gou, Z. Xu, and C. Wu, "Data interpreter: An llm agent for data science," 2024. [Online]. Available: https://arxiv.org/abs/2402.18679

[6] S. Guo, C. Deng, Y. Wen, H. Chen, Y. Chang, and J. Wang, "Ds-agent: Automated data science by empowering large language models with case-based reasoning," 2024. [Online]. Available: https://arxiv.org/abs/2402.17453

[7] U. B. Karli, J.-T. Chen, V. N. Antony, and C.-M. Huang, "Alchemist: Llm-aided end-user development of robot applications," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 361–370. [Online]. Available: https://doi.org/10.1145/3610977.3634969

[8] Y. Ge, Y. Dai, R. Shan, K. Li, Y. Hu, and X. Sun, "Cocobo: Exploring large language models as the engine for end-user robot programming," in *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2024, pp. 89–95.

[9] S. Gathani, Z. Liu, P. J. Haas, and Ç. Demiralp, "What-if analysis for business users: Current practices and future opportunities," *Conference on Human Factors in Computing Systems*, 2022.

[10] Y. Zhang, Q. Jiang, X. XingyuHan, N. Chen, Y. Yang, and K. Ren, "Benchmarking data science agents," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 5677–5700. [Online]. Available: https://aclanthology.org/2024.acl-long.308/

[11] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, S. W. tau Yih, D. Fried, S. Wang, and T. Yu, "Ds-1000: A natural and reliable benchmark for data science code generation," *ArXiv*, vol. abs/2211.11501, 2022.

[12] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3491101.3519665

[13] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, "Github copilot ai pair programmer: Asset or liability?" *J. Syst. Softw.*, vol. 203, no. C, Sep. 2023. [Online]. Available: https://doi.org/10.1016/j.jss.2023.111734

[14] M. Q. Feldman and C. J. Anderson, "Non-expert programmers in the generative ai future," in *Proceedings of the 3rd Annual Meeting of the Symposium on Human-Computer Interaction for Work*, ser. CHIWORK '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3663384.3663393

[15] P. YM, V. Ganesan, D. K. Arumugam, M. Gupta, N. Shadagopan, T. Dixit, S. Segal, P. Kumar, M. Jain, and S. Rajamani, "Pwr: Exploring the role of representations in conversational programming," 2023. [Online]. Available: https://arxiv.org/abs/2309.09495

[16] H. Yan, T. D. Latoza, and Z. Yao, "Intelliexplain: Enhancing conversational code generation for non-professional programmers," 2024. [Online]. Available: https://arxiv.org/abs/2405.10250

[17] K. Ferdowsi, J. Williams, I. Drosos, A. D. Gordon, C. Negreanu, N. Polikarpova, A. Sarkar, and B. Zorn, " COLDECO: An End User Spreadsheet Inspection Tool for AI-Generated Code ," in *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2023, pp. 82–91. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/VL-HCC57772.2023.00017

[18] G. Bansal, T. Wu, J. Zhou, R. Fok, B. Nushi, E. Kamar, M. T. Ribeiro, and D. Weld, "Does the whole exceed its parts? the effect of ai explanations on complementary team performance," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445717

[19] J. Y. Bo, S. Wan, and A. Anderson, "To rely or not to rely? evaluating interventions for appropriate reliance on large language models," 2024. [Online]. Available: https://arxiv.org/abs/2412.15584

[20] R. Fok and D. S. Weld, "In search of verifiability: Explanations rarely enable complementary performance in ai-advised decision making," *AI Magazine*, vol. 45, no. 3, pp. 317–332, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aaai.12182

[21] P. R. Laughlin and A. L. Ellis, "Demonstrability and social combination processes on mathematical intellective tasks," *Journal of Experimental Social Psychology*, vol. 22, no. 3, pp. 177–189, 1986. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022103186900223

[22] N. Jain, R. Kwiatkowski, B. Ray, M. K. Ramanathan, and V. Kumar, "On mitigating code llm hallucinations with api documentation," 2024. [Online]. Available: https://arxiv.org/abs/2407.09726

[23] J. Yang, A. Prabhakar, K. R. Narasimhan, and S. Yao, "Intercode: Standardizing and benchmarking interactive coding with execution feedback," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. [Online]. Available: https://openreview.net/forum?id=fvKaLF1ns8

[24] A. Ni, S. Iyer, D. Radev, V. Stoyanov, W. tau Yih, S. I. Wang, and X. V. Lin, "Lever: Learning to verify language-to-code generation with execution," 2023. [Online]. Available: https://arxiv.org/abs/2302.08468

[25] L. Analytics, "Marketing mix model: Insights and optimization," 2019, accessed: 2025-01-04. [Online]. Available: https://www.latentview.com/wp-content/uploads/2019/08/Marketing-Mix-Model.pdf

[26] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," *ACM Comput. Surv.*, vol. 43, no. 3, Apr. 2011. [Online]. Available: https://doi.org/10.1145/1922649.1922658

[27] J. Good, K. Howland, and K. Nicholson, "Young people's descriptions of computational rules in role-playing games: An empirical study," in *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, ser. VLHCC '10. USA: IEEE Computer Society, 2010, p. 67–74. [Online]. Available: https://doi.org/10.1109/VLHCC.2010.18

[28] Prolific, "Prolific," https://www.prolific.com, 2014.

[29] R. A. Fisher, "Statistical methods for research workers," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1970, pp. 66–70.