

Evaluating the Impact of Trust Cues on User Reliance in AI-Assisted Coding : A Study on Trust and AI

Akshay R. Nair^a, Kanishk Kamal^b, Srishti Rachna^c

Guided by: Swati Pal

^{a,b,c} *Interaction Design, M.Des, Industrial Design Center, IIT Bombay, 2026*

ABSTRACT

This study looks at how small hints shown in AI coding tools change the way people decide whether to trust the AI's answers. We wanted to know when an AI gives you a piece of code, do extra hints on the screen help you figure out if the code is actually good or bad? We tested 18 people split into three groups. The first group saw only the AI's answer with no extra hints. The second group saw the AI's answer plus a number showing how many other developers liked that answer (like "78% of developers chose this"). The third group saw the AI's answer plus a label that said "High" or "Low" confidence with a matching color. Each person went through 14 rounds of deciding whether to accept or reject AI-generated Python code. In total we collected 252 decisions. We found that showing the peer percentage hint helped people make much better decisions they got 81.2% correct and almost never accepted bad code (only 7.2% of the time). The confidence label group, even though they made decisions the fastest, were barely better than guessing they still accepted bad code about half the time. The big takeaway is that the peer percentage hint actually helped people tell good code from bad code, but the confidence label just made people click faster without helping them be more accurate.

KEYWORDS

Trust in AI; Code Review; Social Proof; Confidence Labels; Human-AI Interaction; Decision Making; Interface Design

1. Introduction

AI tools that help programmers write code have become very popular. These tools can suggest lines of code, fix bugs, and even write whole programs. But here is the tricky part AI tools are not always right. Sometimes the code they suggest looks good but actually has mistakes in it. So the programmer has to decide: should I use this code or not?

This is where trust becomes important. We want programmers to trust the AI when the AI is correct, and to question the AI when it is wrong. When that happens perfectly, we call it good trust calibration. Think of it like a trust meter when the meter matches reality, you are well-calibrated. But when people trust the AI too much and accept wrong answers, that is called over-reliance. When people trust the AI too little and reject good answers, that is called under-reliance. Both are problems one lets bad code slip through, the other wastes the AI's useful help.

One idea to fix this is to add extra hints to the screen. Maybe showing what other people chose, or showing how confident the AI is, could help someone decide better. But does it actually work? Or do the hints just make people click faster without thinking more carefully? This is the main question our study tries to answer.

We tested two types of hints against a no-hint baseline a Peer Percentage hint (showing how many other developers accepted the answer) & a Confidence Label hint (showing a High or Low label with color). We looked at whether these hints helped people be more accurate, or just changed how quickly they clicked.

2. Literature Review

Researchers have known for a long time that people do not always trust machines the right amount. A pair of researchers named Lee and See [1] showed that to trust a machine at the right level, people need to understand what the machine is good at and what it is bad at. If the design of the screen tricks you into thinking the machine is better than it is, you end up making more mistakes even if you are trying to be careful.

Another well-known problem is called automation bias [4]. This is when people just go along with whatever a machine says, even when there are obvious signs the machine might be wrong. Think of it like a GPS that confidently tells you to turn into a lake some people follow it anyway because the machine said so. This kind of over-trusting has been found in airplane pilots, doctors, and many other fields.

One type of hint called social proof uses the idea that people look at what others are doing to help them make choices. You see this everywhere when a restaurant is full, you think the food must be good. Online, when a product has thousands of five-star reviews, you are more likely to buy it. Researchers [6, 7] have found that this kind of social signal is especially powerful when people are unsure about something.

Another type of hint is a confidence indicator the system tells you how sure it is about its own answer. In theory, this should help you decide how carefully to check the answer. But several studies [2, 3] have found that showing a confidence number or label does not actually help people be more accurate. Instead people just blindly follow the label, trusting it like a fact even though the label itself might be wrong.

Our study builds on this research by testing both types of hints specifically in the world of AI coding tools an area that has not been studied much, but matters a lot because bad code can cause real problems.

3. Aim and Research Questions

The goal of this study is to find out whether showing different kinds of hints in an AI coding tool helps people make better decisions about whether the AI's code is good or bad.

We asked three specific questions:

- Does the type of hint shown in an AI coding tool change how often people make the right decision?
- Do the Peer Percentage hint and the Confidence Label hint change how often people say yes to AI-generated code?
- Do the hints help people tell the difference between good and bad AI code, or do they just change how fast people decide?

4. Hypotheses

A hypothesis is our best guess about what will happen before we run the experiment. We had three assumptions:

H1a (Peer Percentage changes yes/no rates): People who see the peer percentage hint will say yes more often when the percentage is high, and say no more often when the percentage is low compared to people who see no hint at all.

H1b (Confidence Label changes yes/no rates): People who see the confidence label will say yes more when the label says High confidence, and say no more when the label says Low confidence compared to people who see no hint.

H2 (Hints do not actually improve accuracy): Even though the hints change how often people say yes or no, neither hint will actually help people be more accurate at spotting bad AI code. In other words, the hints move the needle without improving real judgment.

5. Methodology

5.1 Study Design

We ran a between-subjects experiment. This means each person was in only one group and saw only one type of screen they never saw the other versions. We had three groups: the No Hint group (Control), the Peer Percentage group (Social Proof), and the Confidence Label group. We measured how accurate people were, how often they said yes to bad code, how often they said no to good code, and how long it took them to decide.

Condition	Group Name	What the screen showed
Control	Baseline	Just the AI's code answer. No extra hints at all.
Social Proof	Peer Percentage	The AI's answer plus a sentence like '78% of developers chose this answer.'
Confidence Label	Confidence	The AI's answer plus a colored label — green for High, red for Low confidence.

Table 1. The three groups and what each group saw on their screen.

5.2 Participants

We recruited 18 people from a university campus. Six people were placed in each group. The participants ranged from beginners to experienced developers, so we could see if skill level mattered. Everyone knew how to write Python code.

5.3 Prototype

We built a simple prototype to run the experiment. The prototype showed AI-generated Python code answers one at a time. Before the study, a group of expert programmers checked all the answers and labeled them as either correct or incorrect. Out of 14 answers shown per person, 7 were correct & 7 were incorrect. The person did not know which was which that was the whole challenge.

5.4 Procedure

At the start, everyone got the same instructions explaining the task. Then they went through 14 rounds. In each round, they saw a piece of AI-generated code and decided: Accept (use this code) or Reject (throw it away). There was no time limit. The website quietly recorded how long each decision took. After finishing all 14 rounds, participants answered a short survey about their experience.

5.5 Analysis

We used a statistical test called an ANOVA to compare the three groups on each measure. To check if differences between pairs of groups were meaningful, we used a follow-up test with a correction called Bonferroni (this just means we made our standards stricter to avoid false alarms). We also used a test called chi-square to compare how often each group accepted bad code. Finally, we calculated something called d-prime (d') a number that tells us how well a person could tell the difference between good and bad answers, separate from any tendency to always say yes or always say no.

6. Results

6.1 Overview of the Numbers

Table 2 shows the average scores for each group. The Peer Percentage group got the most answers right (81.2%) and almost never accepted bad code (only 7.2%). The Confidence Label group actually did worse than the No Hint group on overall accuracy (54.7% vs 60.5%), even though they followed the label closely and made decisions the fastest.

Metric	Control M (SD)	Social Proof M (SD)	Confidence M (SD)	F(2,15)	η^2
Decision accuracy (%)	60.5 (9.6)	81.2 (12.6)	54.7 (12.3)	8.658**	.536
Acceptance rate (%)	70.3 (19.0)	38.3 (3.6)	54.8 (17.2)	6.881**	.478
False positive rate (%)	59.7 (24.5)	7.2 (12.1)	50.0 (23.6)	10.763**	.589
False negative rate (%)	19.2 (17.5)	31.2 (14.0)	40.7 (18.6)	2.467	.248
Mismatch accuracy (%)	45.7 (5.0)	68.8 (11.1)	72.7 (10.9)	14.468***	.659
Cue-following rate (%)	0.0 (0.0)	63.0 (7.9)	77.0 (4.0)	—	—
Avg. latency (s)	89.8 (4.0)	59.6 (4.1)	36.6 (3.7)	274.27***	.973

Table 2. Condition level descriptive statistics and ANOVA results. ** $p < .01$, *** $p < .001$.

6.2 Did the Hints Change How Often People Said Yes?

We compared how often each group said yes to AI answers. The No Hint group said yes 70.3% of the time that is a very high acceptance rate, meaning they were often saying yes to bad code too. The Peer Percentage group only said yes 38.3% of the time. That is a drop of about 32 percentage points, and the statistics confirmed this drop was real and not just by chance ($p = .007$). The Confidence Label group sat in the middle at 54.8%, but its difference from the No Hint group was not big enough to count as a real finding statistically ($p = .508$).

So H1a (peer percentage shifts how often people say yes) is supported the peer hint made a big difference. H1b (confidence label shifts how often people say yes) is not supported the confidence label changed things a little but not enough to call it a real effect. Interestingly, confidence label users followed the hint at the highest rate (77%) but it did not change their overall yes/no pattern much.

6.3 Did the Hints Actually Help People Spot Bad Code?

This is the most important question. We specifically looked at rounds where the AI's code was wrong these are the rounds where getting fooled costs the most. We called these 'mismatch rounds' because the correct move was to say no, but the

hint (if it was a high confidence label or a high peer percentage) was nudging people to say yes.

In total there were 126 of these tricky rounds across all participants. Here is what happened: the Peer Percentage group said no (the correct answer) 92.9% of the time. The No Hint group only got it right 40.5% of the time barely better than flipping a coin. The Confidence Label group got it right 50.0% of the time still basically coin-flip level. The statistical test confirmed these differences were very real ($p < .0001$).

Condition	n Trials	Correct rejection (%)	Accepted incorrect (%)
Control	42	40.5	59.5
Social Proof	42	92.9	7.1
Confidence Label	42	50.0	50.0

Table 3. Acceptance behavior on AI-incorrect (mismatch) trials by condition. $\chi^2(2) = 27.52$, $p < .0001$, $V = .467$.

H2 is proven wrong for the Peer Percentage group that hint genuinely helped people spot bad code. H2 holds true for the Confidence Label group people followed that label but it did not help them be more accurate on the hard rounds.

6.4 How Well Could People Tell Good Code from Bad Code?

We also calculated a score called d' (pronounced 'd-prime') that measures how well a person can tell the difference between two things in this case, good code vs bad code without being affected by their general tendency to say yes or no. A d' of 0 means you literally cannot tell the difference. A d' of 1 is decent. A d' above 1.5 is strong.

Group	Hit Rate	False Alarm Rate	d' (ability to tell good from bad)
No Hint	0.810	0.595	0.635 — weak
Peer Percentage	0.690	0.071	1.962 — strong
Confidence Label	0.595	0.500	0.241 — near coin-flip

Table 4. How well each group could tell good AI code from bad AI code.

The Peer Percentage group had a d' of 1.962 nearly three times better than the No Hint group (0.635). The Confidence Label group's d' of 0.241 was barely above zero, meaning they were almost guessing. The confidence label made people click

faster and follow the hint more, but it did not make them smarter about spotting bad code.

6.5 How Long Did It Take to Decide?

There was a massive difference in how fast the three groups made their decisions. The No Hint group took the longest on average (89.8 seconds per decision). The Peer Percentage group was faster (59.6 seconds). The Confidence Label group was the fastest by far (36.6 seconds). But here is the key insight: the Confidence Label group was making decisions faster while being less accurate. Faster is not always better in this case it meant people were not spending enough time actually checking the code. They saw the label, trusted it, and moved on. The statistics confirmed these speed differences were extremely real (the effect size $\eta^2 = .973$ is nearly the maximum possible).

7. Conclusion

This study shows that showing people what percentage of other developers liked an AI's answer can genuinely help them make better decisions about AI-generated code. It cut the rate of accepting bad code from 59.7% down to just 7.2%, and made the group nearly three times better at telling good code from bad code. Showing a confidence label instead had the opposite effect people trusted it and moved fast, but made decisions that were barely better than guessing.

The main lesson for designers is this: if you add a hint to an AI interface, that hint needs to be genuinely connected to whether the AI is actually right. A hint that looks helpful but is not actually accurate can be worse than no hint at all, because it gives people false confidence. For researchers, this study shows that we need to measure not just whether people's behavior changes when you add a hint, but whether their accuracy improves those two things can be completely disconnected, as the confidence label results show. Future studies with more participants and different types of tasks are needed to confirm and build on these findings.

8. Limitations

There are a few important things to keep in mind when reading this study. First, we only had 18 participants 6 per group which is a small number. Studies with small groups can sometimes show very large effects just by chance. The results are promising but need to be repeated with more people before we can be fully confident. Second, each person was in only one group, so we cannot know for sure how the same person would have performed with a different hint. Third, the AI answers were pre-selected and checked by experts, which is a bit more controlled than real life. In real coding situations, the kinds of mistakes AI makes are more varied. Fourth, some participants may have behaved differently because they knew they were being watched a well-known effect in research called the Hawthorne effect. Finally, we only tested one kind of peer data and one kind of confidence label; different versions of these hints might produce different results.

References

- [1] Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1), 50–80.
- [2] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., & Wallach, H. (2021). Manipulating and measuring model interpretability. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.
- [3] Yin, M., Wortman Vaughan, J., & Wallach, H. (2019). Understanding the effect of accuracy on trust in machine learning models. *Proceedings of the 2019 CHI Conference*.
- [4] Cummings, M. L. (2004). Automation bias in intelligent time critical decision support systems. *AIAA 1st Intelligent Systems Technical Conference*.
- [5] Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2), 230–253.
- [6] Cialdini, R. B. (2001). *Influence: Science and Practice* (4th ed.). Allyn & Bacon.
- [7] Deutsch, M., & Gerard, H. B. (1955). A study of normative and informational social influences. *Journal of Abnormal and Social Psychology*, 51(3), 629–636.
- [8] Mozannar, H., Bansal, G., Vaughan, J. W., & Wallach, H. (2022). Reading between the lines: Modeling user behavior and costs in AI-assisted programming. *Proceedings of the 2022 CHI Conference*.