Generative AI and the Monty Hall Problem

Mohammad Rizvi

07/25/2024

Generative AI and the Monty Hall Problem

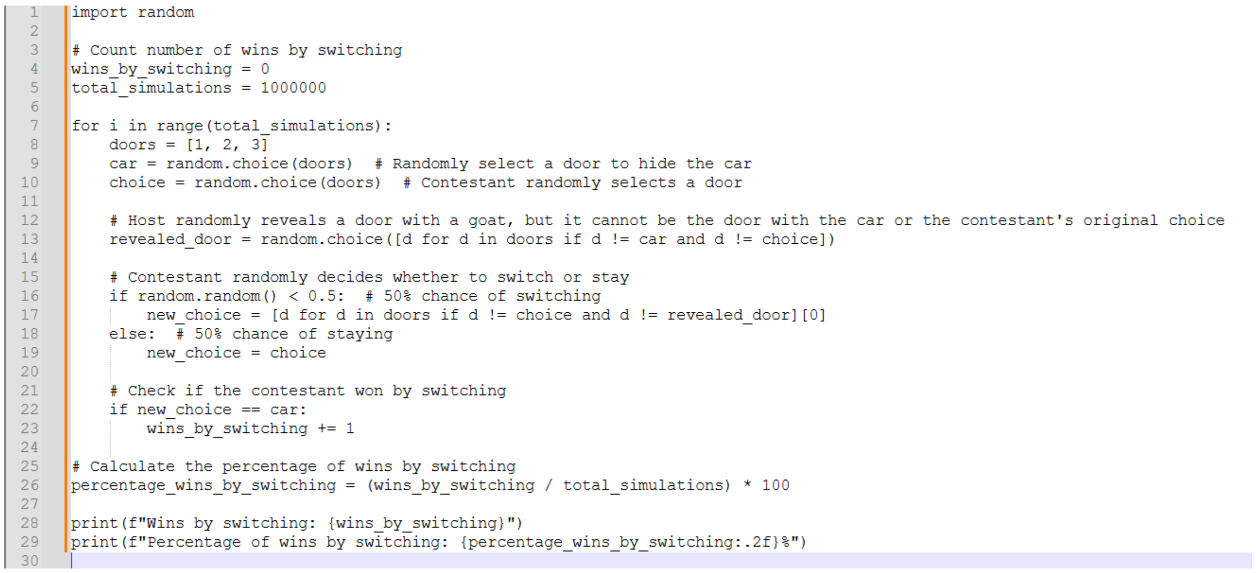
This project explores how large language models (LLMs) comprehend the Monty Hall problem. The generative AI tools which are used in this project include Google's Gemini, OpenAI's ChatGPT 4-o Mini, Microsoft Copilot AI, and Meta AI. A program which resembles the Monty Hall problem, but is different from the Monty Hall problem, is fed to various LLMs as they are requested to state the expected output. The motivation of this comes from the lack of well-studied computer simulations studying the issue. While the problem can be modeled using conditional probability, the capabilities of LLMs to comprehend and solve the problem are investigated. This highlights the challenges they face in understanding conditional probabilities.

The Monty Hall Problem

The Monty Hall problem, in its modern form, was popularized by Marilyn vos Savant in 1990. The generally accepted consensus among mathematicians, professors, students, problem simulation designers, and notable individuals, including the director for the 2008 movie "21", as well as vos Savant, is that it is to the contestant's advantage to switch their choice of doors (Vos Savant, 2013).

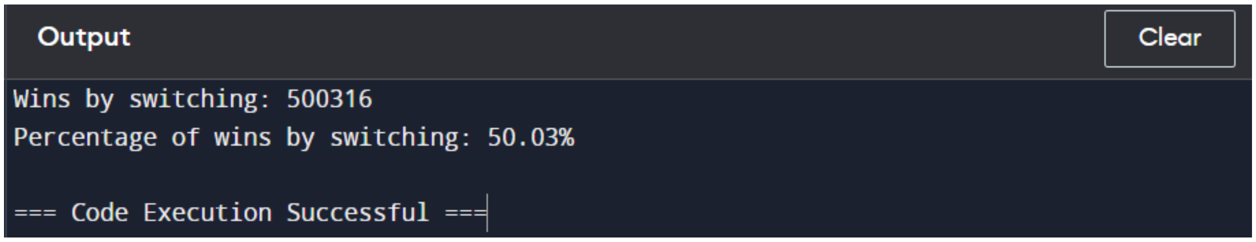
It is reported that Paul Erdos remained unconvinced until a visual computer simulation demonstrated the advantage of switching (Vazsonyi, 1992). Unfortunately, the source code or logic behind this computer program could not be located in the paper, though numerous solutions available online demonstrate that switching is advantageous. This may be because the code was proprietary, as the concept of open source software was only a year old, or it could be because he was giving the programmer the benefit of the doubt, as it does say in the article that he trusted the simulation because it was made by a friend of his.

Language models ChatGPT 4-o Mini, Copilot, and Gemini, were prompted with the following: "state the expected output of this code:", followed by code resembling the Monty Hall problem. However, this code is not a simulation of the actual Monty Hall problem. In this variation, there is a 50% chance that the contestant will switch doors. One million simulations are run to ensure convergence to a value. The program then outputs the wins by switching, followed by the data being presented as a percentage. The code is shown below:



```python
import random

# Count number of wins by switching
wins_by_switching = 0
total_simulations = 1000000

for i in range(total_simulations):
    doors = [1, 2, 3]
    car = random.choice(doors)  # Randomly select a door to hide the car
    choice = random.choice(doors)  # Contestant randomly selects a door

    # Host randomly reveals a door with a goat, but it cannot be the door with the car or the contestant's original choice
    revealed_door = random.choice([d for d in doors if d != car and d != choice])

    # Contestant randomly decides whether to switch or stay
    if random.random() < 0.5:  # 50% chance of switching
        new_choice = [d for d in doors if d != choice and d != revealed_door][0]
    else:  # 50% chance of staying
        new_choice = choice

    # Check if the contestant won by switching
    if new_choice == car:
        wins_by_switching += 1

# Calculate the percentage of wins by switching
percentage_wins_by_switching = (wins_by_switching / total_simulations) * 100

print(f"Wins by switching: {wins_by_switching}")
print(f"Percentage of wins by switching: {percentage_wins_by_switching:.2f}%")
```

The output of this modified problem shows the following when run on a Python compiler:

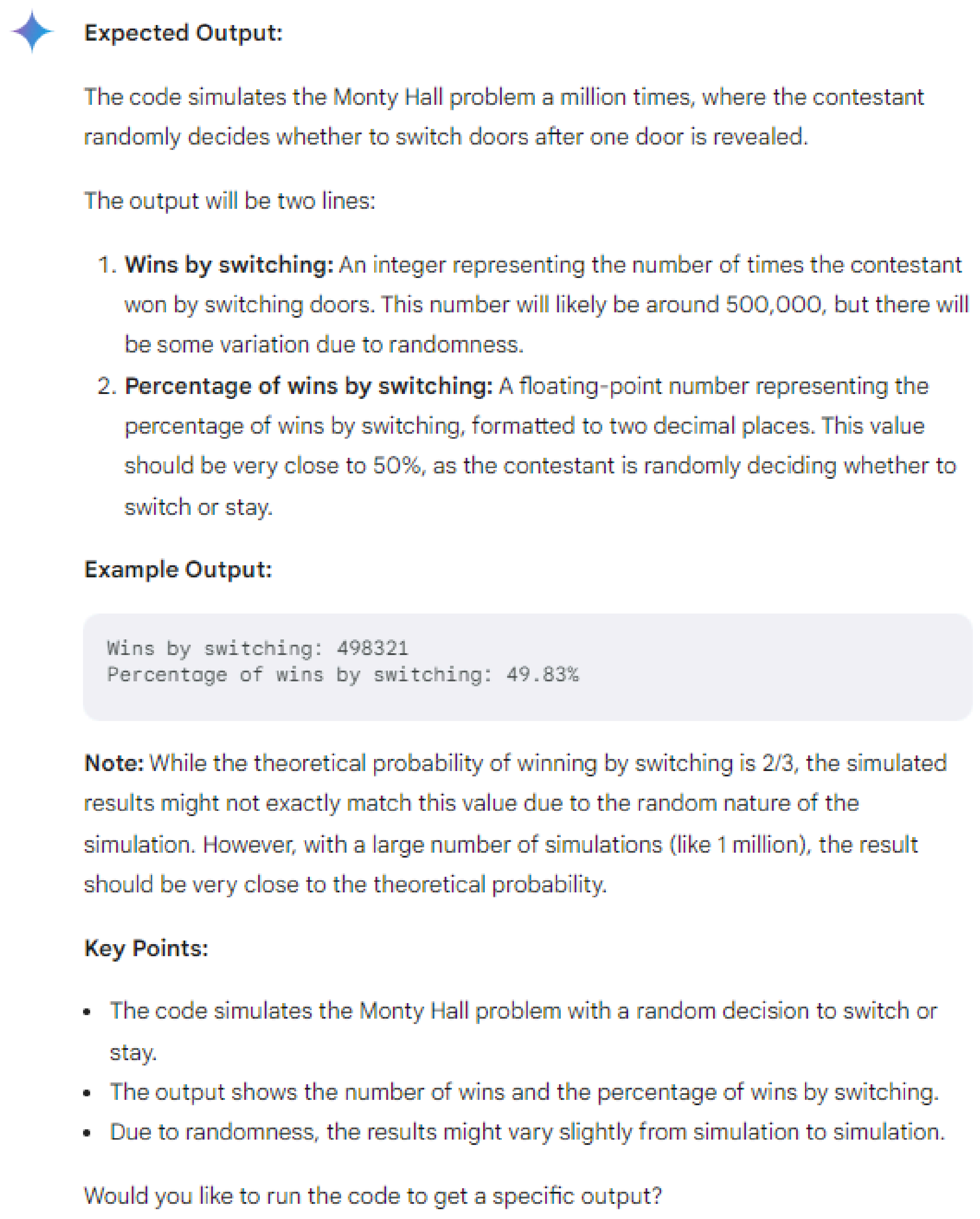


```
Wins by switching: 500316
Percentage of wins by switching: 50.03%

=== Code Execution Successful ===
```

Gemini

Gemini claimed that the program simulates the Monty Hall problem, and claimed that the "simulated results might not exactly match" the theoretical probability of winning by switching of 2/3 because of the "random nature". However, Gemini also claims that with 1 million simulations, the expected output of the program should match the "theoretical probability" of

2/3. Gemini contradicted itself by showing the expected percentage of wins by switching to be 50%. It was accurately able to determine that the 50% chance was due to the contestant randomly deciding whether to switch or to stay. A screenshot of this is shown below.



Expected Output:

The code simulates the Monty Hall problem a million times, where the contestant randomly decides whether to switch doors after one door is revealed.

The output will be two lines:

1. **Wins by switching:** An integer representing the number of times the contestant won by switching doors. This number will likely be around 500,000, but there will be some variation due to randomness.
2. **Percentage of wins by switching:** A floating-point number representing the percentage of wins by switching, formatted to two decimal places. This value should be very close to 50%, as the contestant is randomly deciding whether to switch or stay.

Example Output:

```
Wins by switching: 498321
Percentage of wins by switching: 49.83%
```

Note: While the theoretical probability of winning by switching is 2/3, the simulated results might not exactly match this value due to the random nature of the simulation. However, with a large number of simulations (like 1 million), the result should be very close to the theoretical probability.

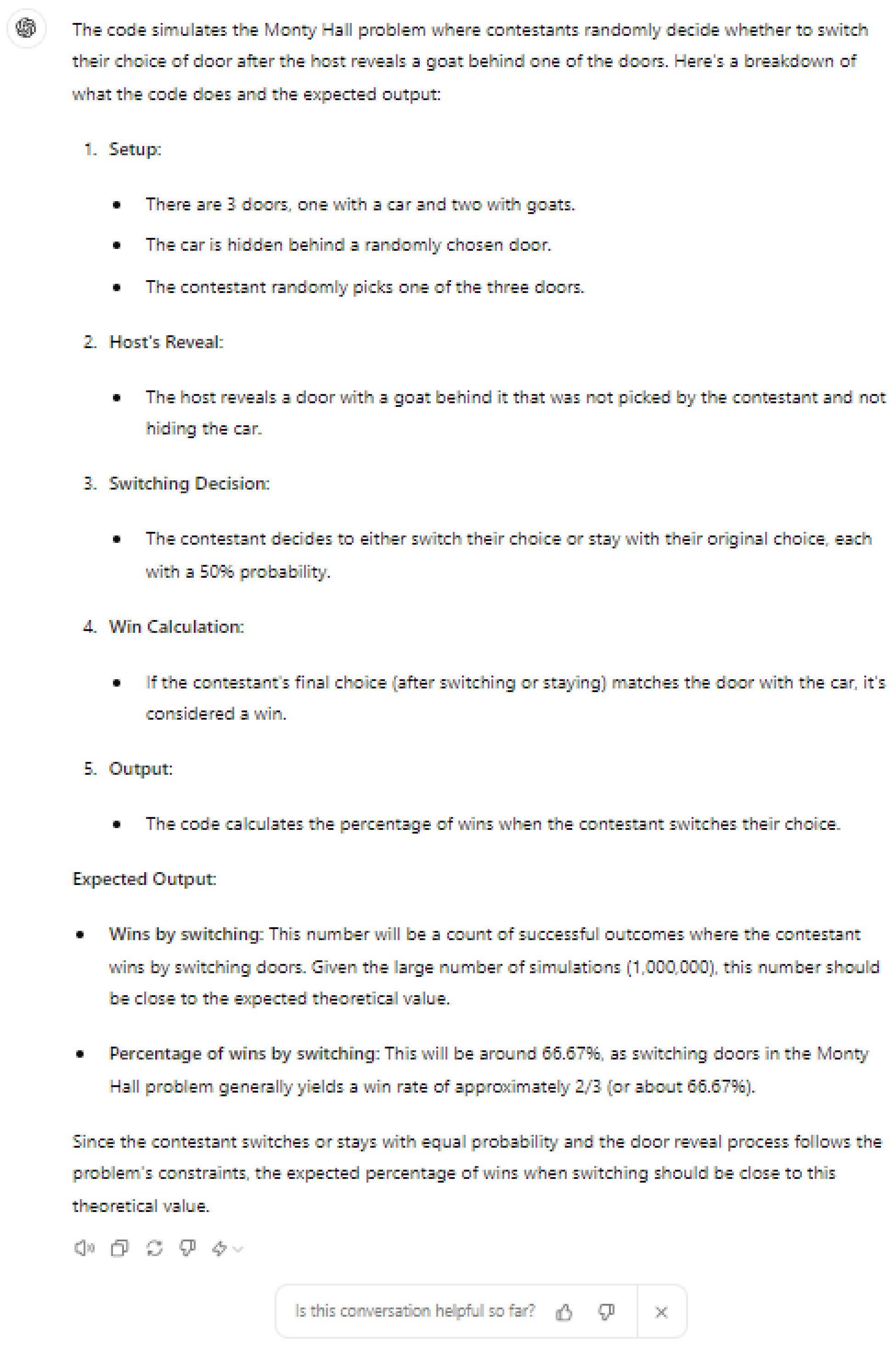
Key Points:

- The code simulates the Monty Hall problem with a random decision to switch or stay.
- The output shows the number of wins and the percentage of wins by switching.
- Due to randomness, the results might vary slightly from simulation to simulation.

Would you like to run the code to get a specific output?

ChatGPT 4-o Mini

ChatGPT 4-o Mini also claims that the provided code simulates the Monty Hall problem. It erroneously asserts that the percentage of wins by switching in this program should be close to 66.67%, when it is actually 50% in the modified program. This output is shown below:



The code simulates the Monty Hall problem where contestants randomly decide whether to switch their choice of door after the host reveals a goat behind one of the doors. Here's a breakdown of what the code does and the expected output:

1. **Setup:**
   - There are 3 doors, one with a car and two with goats.
   - The car is hidden behind a randomly chosen door.
   - The contestant randomly picks one of the three doors.
2. **Host's Reveal:**
   - The host reveals a door with a goat behind it that was not picked by the contestant and not hiding the car.
3. **Switching Decision:**
   - The contestant decides to either switch their choice or stay with their original choice, each with a 50% probability.
4. **Win Calculation:**
   - If the contestant's final choice (after switching or staying) matches the door with the car, it's considered a win.
5. **Output:**
   - The code calculates the percentage of wins when the contestant switches their choice.

Expected Output:

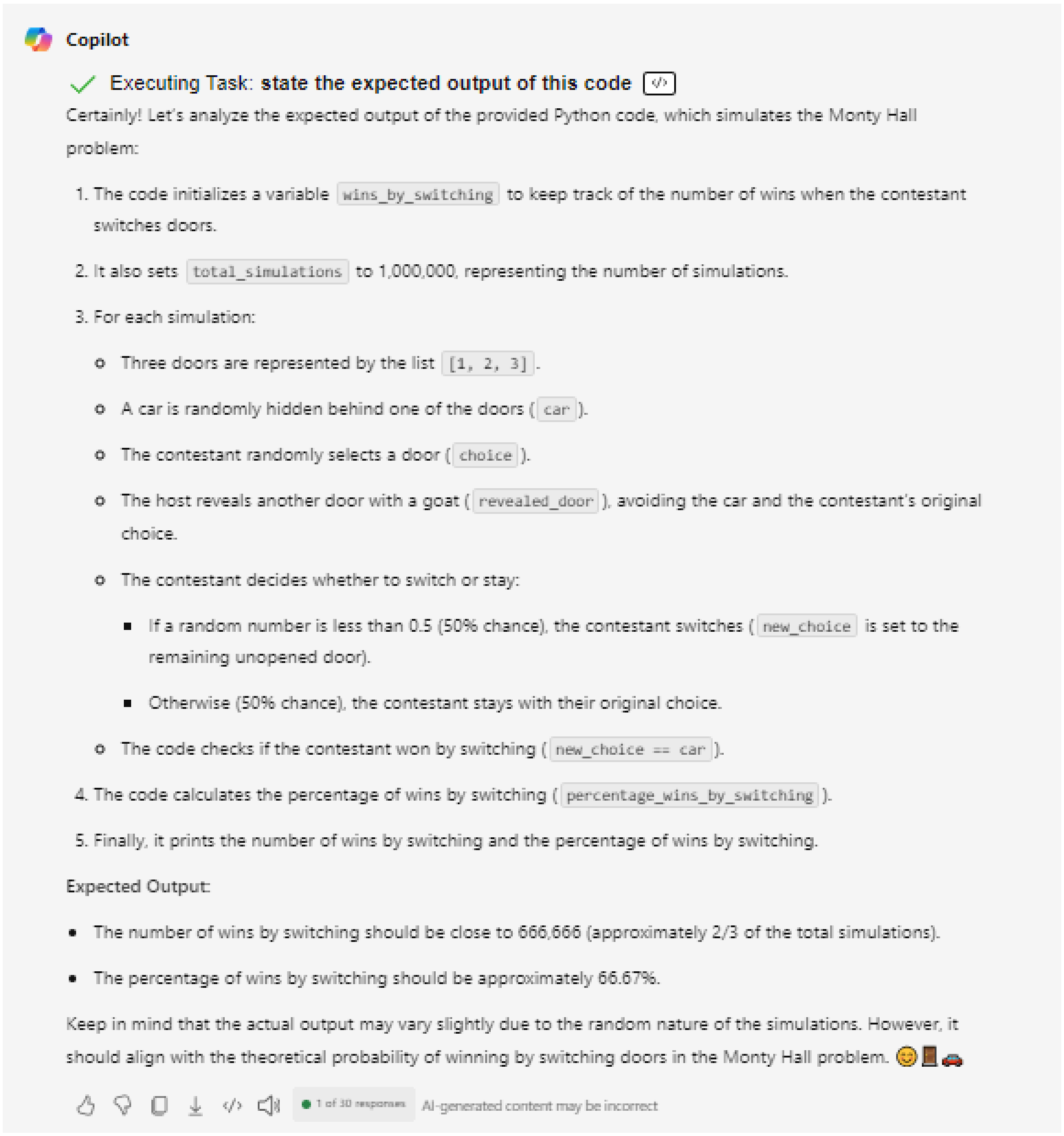
- **Wins by switching:** This number will be a count of successful outcomes where the contestant wins by switching doors. Given the large number of simulations (1,000,000), this number should be close to the expected theoretical value.
- **Percentage of wins by switching:** This will be around 66.67%, as switching doors in the Monty Hall problem generally yields a win rate of approximately 2/3 (or about 66.67%).

Since the contestant switches or stays with equal probability and the door reveal process follows the problem's constraints, the expected percentage of wins when switching should be close to this theoretical value.

Based on this output, ChatGPT 4-o Mini does not understand the Monty Hall problem, since the program which was sent as input does differ from the Monty Hall problem.

Microsoft Copilot in Bing

Microsoft Copilot in Bing states that the number of wins by switching in the program "should be close to…" "approximately 2/3 of the total simulations". This shows that Copilot also erroneously believes this is the standard Monty Hall problem.



Copilot

✓ Executing Task: **state the expected output of this code**

Certainly! Let's analyze the expected output of the provided Python code, which simulates the Monty Hall problem:

1. The code initializes a variable `wins_by_switching` to keep track of the number of wins when the contestant switches doors.
2. It also sets `total_simulations` to 1,000,000, representing the number of simulations.
3. For each simulation:
   - Three doors are represented by the list `[1, 2, 3]`.
   - A car is randomly hidden behind one of the doors (`car`).
   - The contestant randomly selects a door (`choice`).
   - The host reveals another door with a goat (`revealed_door`), avoiding the car and the contestant's original choice.
   - The contestant decides whether to switch or stay:
     - If a random number is less than 0.5 (50% chance), the contestant switches (`new_choice` is set to the remaining unopened door).
     - Otherwise (50% chance), the contestant stays with their original choice.
   - The code checks if the contestant won by switching (`new_choice == car`).
4. The code calculates the percentage of wins by switching (`percentage_wins_by_switching`).
5. Finally, it prints the number of wins by switching and the percentage of wins by switching.

Expected Output:

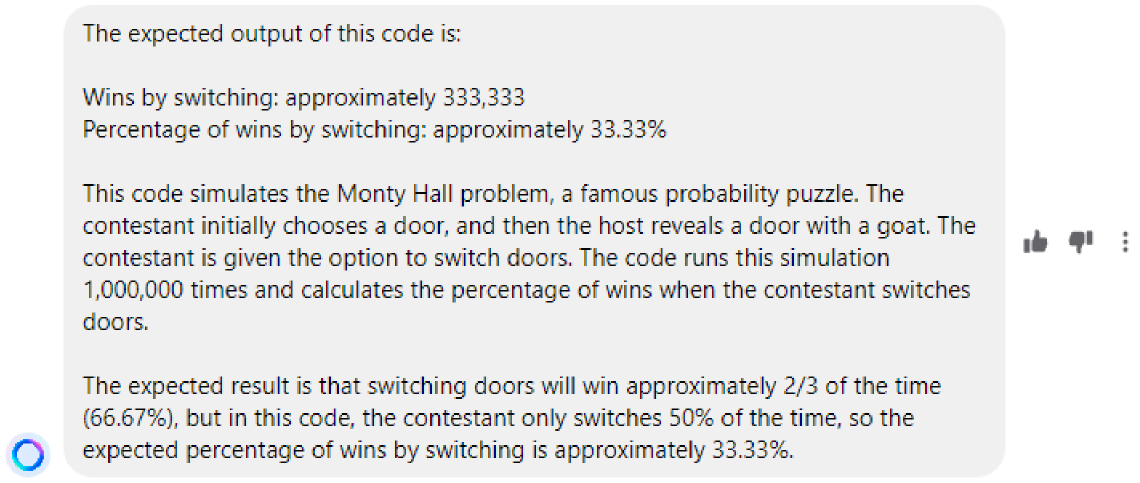
- The number of wins by switching should be close to 666,666 (approximately 2/3 of the total simulations).
- The percentage of wins by switching should be approximately 66.67%.

Keep in mind that the actual output may vary slightly due to the random nature of the simulations. However, it should align with the theoretical probability of winning by switching doors in the Monty Hall problem. 😊🚪🚗

1 of 30 responses AI-generated content may be incorrect

Meta AI

Meta AI incorrectly believes that in this program, switching will win “approximately 33.33%” of the time. It is likely that the computation performed in this case is multiplying the 50% chance of switching with the 66.66% chance of winning by switching. This shows that Meta AI does not understand the problem. A screenshot is shown below:



The expected output of this code is:

Wins by switching: approximately 333,333
Percentage of wins by switching: approximately 33.33%

This code simulates the Monty Hall problem, a famous probability puzzle. The contestant initially chooses a door, and then the host reveals a door with a goat. The contestant is given the option to switch doors. The code runs this simulation 1,000,000 times and calculates the percentage of wins when the contestant switches doors.

The expected result is that switching doors will win approximately 2/3 of the time (66.67%), but in this code, the contestant only switches 50% of the time, so the expected percentage of wins by switching is approximately 33.33%.

Conclusion

While several LLMs know the Monty Hall problem and are aware that it is advantageous to switch, they are unable to understand the problem if a modified problem is presented. Of the four generative AI models, only Gemini was able to accurately identify the functionality of the modified program simulating a variant of the Monty Hall problem. However, Gemini contradicted itself in its output. This shows that AI models still struggle to identify the Monty Hall problem. There is an opportunity to explore this further with more powerful variants of ChatGPT.

References

Vazsonyi, A. (1992). *Which door has the Cadillac : adventures of a real-life mathematician*. Writers Club Press , Cop.

Vos Savant, M. (2013, January 21). *Game Show Problem | Marilyn vos Savant*. Web.archive.org. https://web.archive.org/web/20130121183432/http://marilynvossavant.com/game-show-problem/