# Teaching genAI to Play Diamonds: An Optimizing Strategy

U Harshitha

March 26, 2024

## 1 Introduction

The Diamond Card Game is a two-player bidding game where players compete to win diamond cards, each associated with a numeric value ranging from 1 to 13. The game is played over 13 rounds, with players taking turns to bid on each diamond card as it is revealed from a deck. The bidding process involves players selecting a bid value corresponding to the numeric value of the card, and the player with the highest bid wins the card. However, once a card has been bid on by one player, it cannot be bid on again by either player. At the end of the game, players tally their scores based on the sum of the numeric values of the diamond cards they have won, with the player accumulating the highest score being declared the winner.

Strategic gameplay in the Diamond Card Game revolves around assessing the value of each diamond card relative to its numeric value and considering the bidding behavior of the opponent. Players must decide when to bid aggressively to secure high-value cards and when to bid conservatively to avoid overcommitting bids. Additionally, players can adapt their bidding strategies based on the progression of the game and the cards that have already been played. Successful players balance risk and reward, leveraging their understanding of probabilities and opponent tendencies to outmaneuver their opponent and emerge victorious in this engaging and dynamic bidding game.

## 2 Objective

The objective is to develop a computer program capable of playing Diamonds effectively by employing an optimizing strategy. This involves defining the rules of the game, interacting with genAI to teach it the game, discussing and implementing a strategy, and evaluating its performance through gameplay.

While achieving this we try to understand how AI responds to different prompts and its comprehention capabilities along with its coding skills.

## 3 Methodology

Understanding and implementing the Diamonds game began with asking the AI about its familiarity with the game and providing it with some hints. Though initially, there was some confusion, this helped lay a foundation for further exploration. I then focused on defining the game rules, using several prompts to ensure the AI grasped the mechanics thoroughly. To solidify its understanding, I engaged the AI with questions and scenarios related to the game.

Moving on to coding, I first requested a program where both the user and the AI provide inputs. Through a series of iterations and error corrections, I eventually arrived at a functional program. Transitioning to playing against the computer, I initially worked with random selections but gradually guided the AI to understand optimal strategies for card selection and gameplay. This process involved iterative conversations and testing of various strategies to refine the AI's decision-making abilities. Through this collaborative effort, I were able to develop a comprehensive understanding of the game and implement effective gameplay strategies.

## 4 Reflections on Conversation with genAI & Prompting

Engaging with genAI proved to be an enlightening journey that underscored the complexities of teaching and learning within an AI system. At the outset, there was some initial difficulty in conveying the intricacies of the Diamonds game, and genAI struggled to grasp its rules and dynamics. However, through patient guidance and persistent prompting, I witnessed a remarkable transformation as genAI gradually absorbed the concepts and nuances of the game.

The experience showed the significance of clear communication and iterative refinement in the teaching process. By breaking down the game mechanics into digestible components and providing incremental guidance, I facilitated genAI's comprehension and improved its learning. This iterative approach allowed genAI to navigate through challenges, adapt its understanding, and ultimately, develop a comprehensive grasp of the game.

Moreover, the process highlighted the static + dynamic nature of human-AI interaction, where collaboration and feedback sometimes gave amazing changes in responses but other times led the GenAI to give the same exact answer with different names or comments. This led to confusion about how AI receives responses and if it's the same machine working behind why there were different responses for the same prompt.

## 5 Reflections on Code that was Generated

The code generated by genAI demonstrated a logical approach to playing Diamonds. It incorporated strategies such as card counting, probability assessment, and risk management. However, there were instances where the code exhibited suboptimal decision-making, indicating areas for improvement.

### 5.1 Snippet of Code: Diamond Collection Strategy

1. Code given after corrections for a two user player game

```python
def play_game():
    deck = initialize_deck()
    random.shuffle(deck)

    hearts_player_score = 0
    clubs_player_score = 0
    hearts_played_cards = set()
    clubs_played_cards = set()

    for round in range(1, 14):
        card = deck.pop(0)

        print("\nRound", round, "- Next diamond card:", card)
        hearts_bid = get_valid_bid("Hearts player, place your bid (1-13):
            ")
        while hearts_bid in hearts_played_cards:
            print("Repeated card")
            hearts_bid = get_valid_bid("Hearts player, place your bid
                (1-13): ")

        hearts_played_cards.add(hearts_bid)

        clubs_bid = get_valid_bid("Clubs player, place your bid (1-13): ")
        while clubs_bid in clubs_played_cards:
            print("Repeated card")
            clubs_bid = get_valid_bid("Clubs player, place your bid
                (1-13): ")
        clubs_played_cards.add(clubs_bid)

        if hearts_bid > clubs_bid:
            hearts_player_score += card
            print("Hearts player wins the bid with a bid of", hearts_bid)
        elif clubs_bid > hearts_bid:
            clubs_player_score += card
```

```
                print("Clubs player wins the bid with a bid of", clubs_bid)
            else:
                print("Both players bid the same amount. The card is
                    discarded.")

        print("\nGame Over!")
        print("Hearts player's total score:", hearts_player_score)
        print("Clubs player's total score:", clubs_player_score)

        if hearts_player_score > clubs_player_score:
            print("Hearts player wins!")
        elif hearts_player_score < clubs_player_score:
            print("Clubs player wins!")
        else:
            print("It's a tie!")

    def get_valid_bid(prompt):
        while True:
            bid = input(prompt)
            if bid.isdigit():
                bid = int(bid)
                if 1 <= bid <= 13:
                    return bid
                else:
                    print("Invalid input. Please enter a number between 1 and
                        13.")
            else:
                print("Invalid input. Please enter a number between 1 and 13.")
```

2.Strategy given for computer's choice

```
def computer_choose_bid(remaining_cards, round, diamond_card):
    # Convert remaining cards to a list for indexing
    remaining_cards_list = list(remaining_cards)

    # Adjust the bid based on the value of the diamond card
    if diamond_card > 7:  # If the diamond card is higher than 7
        # Choose from the upper half of remaining cards for higher bids
        valid_bids = [card for card in remaining_cards_list if card > round]
        bid = random.choice(valid_bids) if valid_bids else
            random.choice(remaining_cards_list)
    elif diamond_card < 7:  # If the diamond card is lower than 7
        # Choose from the lower half of remaining cards for lower bids
        valid_bids = [card for card in remaining_cards_list if card <= round]
        bid = random.choice(valid_bids) if valid_bids else
            random.choice(remaining_cards_list)
    else:  # If the diamond card is 7, choose randomly
        bid = random.choice(remaining_cards_list)

    return bid
```

## 6 Analysis

Playing against genAI revealed both its strengths and weaknesses. While genAI demonstrated proficiency in understanding prompts and responding accordingly it did fail in cases of remembering previously stated rules and struggled in correcting logical errors at times while trying to get the code. Also it repetedly made the same error until corrected manually.

A difference in the results produced by different GenAI tools was also observed while Gemini often went with an object oriented approach ChatGPT used the functions based one. Also I felt that the understanding capabilities of ChatGPT was better and more clear prompts were needed for explaining Gemini.

## 7 Conclusion & Path Forward

In conclusion, teaching genAI to play Diamonds presented challenges and opportunities for learning. Through iterative conversations and code generation, genAI developed a foundational understanding of the game and demonstrated promising capabilities. Moving forward, keeping the learnings from these in mind, we can redefine how and where we use GenAI for future projects