BULLS AND COWS GAME WITH ENTROPY

A Project report submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF SCIENCE

IN

APPLIED DATA SCIENCE

Submitted by

SAI SATWIK YARAPOTHINI (UFID : 56927137)

CONTENTS

1. PROBLEM STATEMENT

1.1 Project Overview

The **Bulls and Cows Game with Entropy** combines the engaging nature of a classic guessing game with the mathematical principles of **Information theory**. This project not only entertains but also provides a practical application of entropy, a measure of uncertainty, in a dynamic and interactive environment.

The game challenges the player to guess a secret 4-digit number, generated such that all digits are unique. After each guess, the player receives feedback in the form of:

- **Bulls:** The number of digits that are correct and in the correct position.
- **Cows:** The number of digits that are correct but in the wrong position.

To enhance the traditional gameplay, the concept of entropy is incorporated where the Entropy dynamically calculates the remaining uncertainty based on the number of valid possible states (remaining guesses) and also provides feedback from each guess reduces the set of possible valid states, entropy decreases, reflecting the narrowing uncertainty. This combination of classic gameplay and mathematical modelling makes the project both engaging and academical.

1.2 Problem Statement

The **Bulls and Cows Game with Entropy** project involves developing an Interactive web-based application that combines the classic guessing game with principles of information theory. The game challenges players to guess a secret 4-digit number with unique digits, providing feedback in terms of:

- **Bulls:** The number of digits that are correct and in the correct position.
- **Cows:** The number of digits that are correct but in the wrong position.

This project adheres to the standard rules of the Bulls and Cows game while introducing the concept of **Entropy** to dynamically quantify the uncertainty at each step. The application features: Real-time feedback for each guess, User-friendly interface, Visualizations to track entropy reduction as the game progresses.

2. OBJECTIVES

2.1. Primary Objectives

The primary objectives of the **Bulls and Cows Game with Entropy** project are as follows:

1. Game Design

- To design and implement the core mechanics of the classic **Bulls and Cows** game.
- The game should generate a secret 4-digit number with unique digits, and players should receive feedback after each guess in terms of:
    - **Bulls:** Correct digits in the correct position.
    - **Cows:** Correct digits but in the wrong position.
- The gameplay should be dynamic, ensuring real-time feedback to keep the player engaged.

2. Information Theory Application

- To integrate **Entropy**, a key concept in information theory, into the gameplay.
- Entropy should dynamically represent the uncertainty about the secret number based on the number of valid remaining possibilities.
- Demonstrate how feedback from each guess reduces the entropy, narrowing down the set of valid states which adds an analytical layer to the game, highlighting the practical utility of entropy in decision-making.

3. Interactive Interface

- To develop a web-based interface using an appropriate front-end library, providing a smooth and interactive experience for players.
- The interface should allow users to:
    - Enter guesses through input fields.
    - Receive immediate feedback (Bulls, Cows, and Entropy).
    - Restart the game with the click of a button.
- The design should focus on user-friendliness, accessibility, and responsiveness.

4. Visualization

- To enhance the gameplay by providing visual insights into the entropy reduction process.
- Include real-time charts to display the entropy progression, enabling players to see how uncertainty diminishes after each guess.

These objectives aim to create a seamless experience of using an web application, making the **Bulls and Cows Game with Entropy** an Innovative project.

3. REQUIREMENTS FOR GAMEPLAY

3.1. Tools Required

The implementation of the **Bulls and Cows Game with Entropy** requires a combination of hardware and software tools. These tools ensure the successful development, execution, and interactive gameplay experience.

- **Hardware Requirements:**

- A computing device capable of running Python and handling web-based applications.

Laptop: Any standard laptop with at least 4 GB of RAM and a capable processor which supports Python execution and the ability to run web-based frameworks like Streamlit, ReactJS**.**

- Devices with Internet connectivity to install dependencies seamlessly.

- **Software Requirements:**

- **Python Environment:** This project is implemented in Python, and any compatible version of Python 3.x must be installed on the system to play this game. Visual Studio Code is preferred as running Streamlit application on it is smooth.

These tools collectively ensure the smooth functioning of the game, from generating the secret number to providing an interactive user experience with real-time feedback and entropy calculations.

3.2. Steps to Play the Game

The Bulls and Cows Game with Entropy provides a seamless and engaging experience through its interactive web interface. The following steps outline how to play the game, from setup to winning.

- **Setup:**

- **Launch the Game:** The game is launched using the Streamlit interface, which provides a web-based environment for gameplay.
  - Run the command in the terminal which initializes the game and opens the interface in the default web browser and a random 4-digit number with unique digits is generated at the start of the game which is stored internally and remains hidden from the player.

```
streamlit run (Path of the file .py file is stored in)
```

- **Gameplay**

  - **Input Guesses:** Players enter their guesses through an input field provided in the web interface where each guess must be a 4-digit number with unique digits. The interface ensures validation to prevent invalid inputs.
  - **Feedback Mechanism:** After submitting a guess, the game provides immediate feedback:

    Bulls: Number of digits that are correct and in the correct position.

    Cows: Number of digits that are correct but in the wrong position.

  - **Entropy Calculation:** The entropy, representing the uncertainty about the secret number, is calculated dynamically. As invalid possibilities are eliminated with each guess, the entropy decreases, reflecting the narrowing range of potential solutions.

- **Winning**

  - **Victory Condition:** The game ends when the player correctly guesses the secret number, achieving **4 Bulls** (all digits are correct and in the correct position).Upon winning, the interface displays a Congratulatory message with celebratory visuals i.e. Balloons, The secret number which is guessed correctly, Total number of guesses made to solve the game and an Entropy chart.
  - **Restart Option:** After the game ends, players can restart the game with a new secret number by clicking the "Restart Game" button.

These steps ensure an engaging and educational gameplay experience, combining traditional game mechanics with modern interactivity and mathematical insights.

4. GAME RULES

4.1. Rules

The Bulls and Cows Game with Entropy adheres to the standard rules of the classic guessing game. These rules define the mechanics of gameplay.

- **The Secret Number:** At the start of the game, a secret 4-digit number is randomly generated in which each digit in the secret number is **Unique**, ensuring no repeated digits
    - This remains hidden from the player until the game concludes.
- **Input Requirements:** Players must input a 4-digit number as an Input during their every turn where the player's guess must also consist of unique digits**.** The game validates each guess, and players are notified if their input does not meet the criteria.
- **Feedback Mechanism**: After each guess, the game provides feedback based on a comparison of the player's guess with the secret number. This feedback includes two components:

  Bulls: The number of digits in the guess that are both correct and in the correct position.

  Cows: The number of digits in the guess that are correct but in the wrong position.
- **Objective of the Game** The player's goal is to guess the secret number in the **fewest possible attempts**. As guesses are made, the player can use the feedback to systematically narrow down the possibilities.
    - The game ends when the player achieves **4 Bulls**, meaning the entire number has been guessed correctly.

These rules create a strategic and interactive gameplay experience. They encourage players to think logically, interpret feedback effectively, and make informed guesses to achieve the best results.

5. THEOROETICAL INTERPRETATION

5.1 Entropy

Entropy is a fundamental concept in information theory that quantifies the uncertainty or randomness present in a system. It measures the amount of information required to describe the outcome of a random variable.

- **Definition:** Entropy, denoted as H(X), represents the average level of "information," "surprise," or "uncertainty" inherent in the possible outcomes of a random variable X.
- **Shannon Entropy Formula:**

$$H(X) = -\sum p(x) \log_2(p(x))$$

Where: p(x) is the probability of each outcome x. & Logarithmic base 2 ensures the result is measured in bits.

This formula calculates the weighted average of the information content (measured as log2(p(x)) for all possible outcomes of X, weighted by their respective probabilities p(x).

5.2 Entropy in this Context

In the Bulls and Cows Game, Entropy represents the uncertainty about the secret 4-digit number at any given stage of the game. It provides a quantitative measure of how much information is needed to identify the secret number from the remaining possibilities.

Concept: Entropy is calculated based on the number of remaining valid states (possible combinations of digits that could match the secret number). At the beginning of the game, when all possibilities are equally likely, entropy is at its **Maximum**. As feedback (Bulls and Cows) eliminates invalid possibilities, entropy **Decreases** and eventually becomes **Zero** when correct number is guessed by the user.

5.2.1 Entropy Calculation

- **Initial State:** At the start of the game, all 10×9×8×7= 5040 valid combinations of unique 4-digit numbers are equally likely.

  The probability of each state is:

$$p(x) = \frac{1}{5040}$$

  The entropy of the system is calculated as:

$$H(X) = \log_2(5040) \approx 12.29 \,\text{bits}.$$

  This value indicates the amount of information (in bits) needed to determine the secret number and **12.29 bits is the Entropy which is assumed even before the User starts guessing the Secret number**.

- **Subsequent States:** After each guess, feedback (Bulls and Cows) reduces the set of valid states "S".

  The remaining probabilities are recalculated based on the reduced set of possibilities:

$$p(x) = \frac{1}{|S|}$$

  Where |S| is the size of the reduced set S (remaining valid states).

  The entropy is then recalculated as:

$$H(X) = -\sum_{x \in S} p(x) \log_2(p(x))$$

- **Final State:** When the player correctly guesses the secret number:
  - Only one valid state remains (|S|=1), Probability of the correct state: p(x)=1

    Entropy:

$$H(X) = -1 \cdot \log_2(1) = 0 \,\text{bits}.$$

    At this point, there is no uncertainty, as the secret number has been identified. This calculation highlights how feedback systematically reduces uncertainty, demonstrating the practical application of entropy in Decision-making.

6. CODE IMPLEMENTATION

6.1 Coding Languages

6.1.1. Python

- Python is a high-level, interpreted programming language known for its readability, simplicity, and extensive library support. It is widely used for Data analysis, machine learning, web development and more use cases.
- Python serves as the backbone of this project by implementing the following functionalities**:**

  Game Logic: Functions such as `evaluate_guess` and `filter_possible_states` are implemented in Python to define the rules and mechanics of the Bulls and Cows game.

  Entropy Calculations: Python's `math` library is used to calculate entropy dynamically.

  State Management: The `st.session_state` feature of Streamlit ensures that the game's state persists across user interactions.

6.2 Libraries Used

6.2.1 Streamlit

Streamlit is an open-source Python library used to create interactive web applications for data visualization and user interfaces.

- **Usage in this Code:**

  - Handles user input through widgets like `st.text_input` and `st.button`.
  - Displays real-time feedback, including guesses, Bulls, Cows, and entropy values, using `st.success` and `st.info`.
  - Visualizes entropy progression with `st.line_chart`.
  - Ensures session persistence with `st.session_state`.

6.2.2 Random

The `random` library provides functions to generate random numbers in Python.

- **Usage in this Code:**

  - `random.sample(range(10), 4)` generates the secret 4-digit number with unique digits at the start of the game.

6.2.3 Math

The `math` library provides mathematical functions like logarithms and powers.

- **Usage in this Code:**
  - The `math.log2` function is used to calculate **Shannon entropy** in `calculate_entropy`.

6.2.4 Itertools

The `itertools` library provides functions for efficient iteration and combinatorial operations.

- **Usage in this Code:**
  - `itertools.permutations(range(10), 4)` generates all possible 4-digit numbers with unique digits, used to calculate the initial set of valid states.

6.3 Code Implementation and Explanation

Source Code (Python):

```
# Importing Necessary Libraries
import streamlit as st
import random
import math
from itertools import permutations


# Function to generate a secret 4-digit number with unique digits
def generate_secret():
    return random.sample(range(10), 4)


# Function to generate all possible valid 4-digit numbers (no repeated digits)
def all_possible_states():
    return list(permutations(range(10), 4))


# Function to calculate Shannon entropy based on the remaining possible states
def calculate_entropy(possible_states):
    total = len(possible_states)
    if total == 0:  # Handle edge case where no states remain
        return 0
    probabilities = [1 / total] * total
    return -sum(p * math.log2(p) for p in probabilities)
```

```
# Function to evaluate the player's guess and provide feedback (Bulls and
Cows)
def evaluate_guess(secret, guess):
    # Calculate Bulls (correct digit in the correct position)
    bulls = sum(s == g for s, g in zip(secret, guess))
    # Calculate Cows (correct digit in the wrong position)
    cows = sum((secret.count(g) for g in guess)) - bulls
    return bulls, cows


# Function to filter possible states based on feedback from the current guess
def filter_possible_states(possible_states, guess, bulls, cows):
    # Helper function to check if a state matches the feedback
    def feedback_matches(state):
        state_bulls = sum(s == g for s, g in zip(state, guess))
        state_cows = sum((state.count(g) for g in guess)) - state_bulls
        return state_bulls == bulls and state_cows == cows

    # Filter the states that match the feedback
    return [state for state in possible_states if feedback_matches(state)]


# Function to reset the game state variables and restart the game
def restart_game():
    st.session_state.secret = generate_secret()  # Generate a new secret
number
    st.session_state.guesses = []  # Reset list of guesses
    st.session_state.entropies = []  # Reset entropy values
    st.session_state.game_over = False  # Reset game-over status
    st.session_state.possible_states = all_possible_states()  # Reset possible
states

# Function to set the background color of the Streamlit page to black
def set_background_color():
    st.markdown(
        """
        <style>
        body {
            background-color: black;  /* Set background color */
            color: white;  /* Set text color */
        }
        </style>
        """,
        unsafe_allow_html=True,
    )
```

```python
# Main function to run the Streamlit application
def main():
    set_background_color()  # Set custom background color

    # Ensure all required session state variables are initialized
    if "secret" not in st.session_state:
        st.session_state.secret = generate_secret()
    if "guesses" not in st.session_state:
        st.session_state.guesses = []
    if "entropies" not in st.session_state:
        st.session_state.entropies = []
    if "game_over" not in st.session_state:
        st.session_state.game_over = False
    if "possible_states" not in st.session_state:
        st.session_state.possible_states = all_possible_states()

    # Title and introductory instructions
    st.title("Bulls and Cows Game with Entropy")
    st.write(
        "Welcome to the Bulls and Cows game! Can you guess the secret 4-digit
number?"
    )
    st.write("### Instructions:")
    st.markdown(
        """
    - The secret number has 4 digits, each unique.
    - Enter your guesses below.
    - You will receive feedback:
      - **Bulls**: Correct digit in the correct position.
      - **Cows**: Correct digit in the wrong position.
    - The current entropy will indicate the remaining uncertainty.
    """
    )

    # Add a Restart Game button
    if st.button("Restart Game"):
        restart_game()  # Reset the game state
        st.success("Game has been restarted!")  # Notify the user

    # Game logic and feedback loop
    if not st.session_state.game_over:
        # Input field for the player's guess
        guess_input = st.text_input("Enter your guess (4 unique digits):", "")
        if st.button("Submit Guess"):
            try:
                # Parse the input into a list of integers
                guess = list(map(int, guess_input))
                if len(guess) != 4 or len(set(guess)) != 4:  # Validate input
```

```
                    st.error("Invalid guess. Please enter 4 unique digits.")
                else:
                    # Evaluate the guess and provide feedback
                    bulls, cows = evaluate_guess(st.session_state.secret,
guess)
                    st.session_state.guesses.append((guess, bulls, cows))

                    # Update possible states and calculate entropy
                    st.session_state.possible_states = filter_possible_states(
                        st.session_state.possible_states, guess, bulls, cows
                    )
                    entropy =
calculate_entropy(st.session_state.possible_states)
                    st.session_state.entropies.append(entropy)

                    # Display feedback to the player
                    st.success(f"Guess: {guess} | Bulls: {bulls}, Cows:
{cows}")
                    st.info(f"Current Entropy: {entropy:.4f}")

                    # Check if the player has guessed the number correctly
                    if bulls == 4:
                        st.balloons()  # Celebrate the victory using Ballons
                        st.success("Congratulations! You guessed the number!")
                        st.write(f"Secret number: {st.session_state.secret}")
                        st.write(f"Total guesses:
{len(st.session_state.guesses)}")
                        st.subheader("Your Entropy Chart looks like :")
                        st.line_chart(st.session_state.entropies)  # Show
entropy chart
                        st.session_state.game_over = True  # End the game
            except ValueError:
                st.error("Please enter valid digits.")  # Handle invalid input
    else:
        # Notify the user if the game is over
        st.warning("Game over! Click 'Restart Game' to play again.")


# To Run the application
if __name__ == "__main__":
    main()
```

Explanation:

The Bulls and Cows Game with Entropy has four primary functionalities that ensure seamless gameplay and an engaging user experience. Firstly, the **game initializes by generating a random 4-digit secret number with unique digits** and storing it in `st.session_state` for session persistence. All valid combinations of digits are precomputed using `itertools.permutations`, forming the initial set of possible states. Additionally, the interface setup is enhanced with custom background styling for a visually appealing user experience i.e. User can choose his theme in his default browser : Dark or Light.

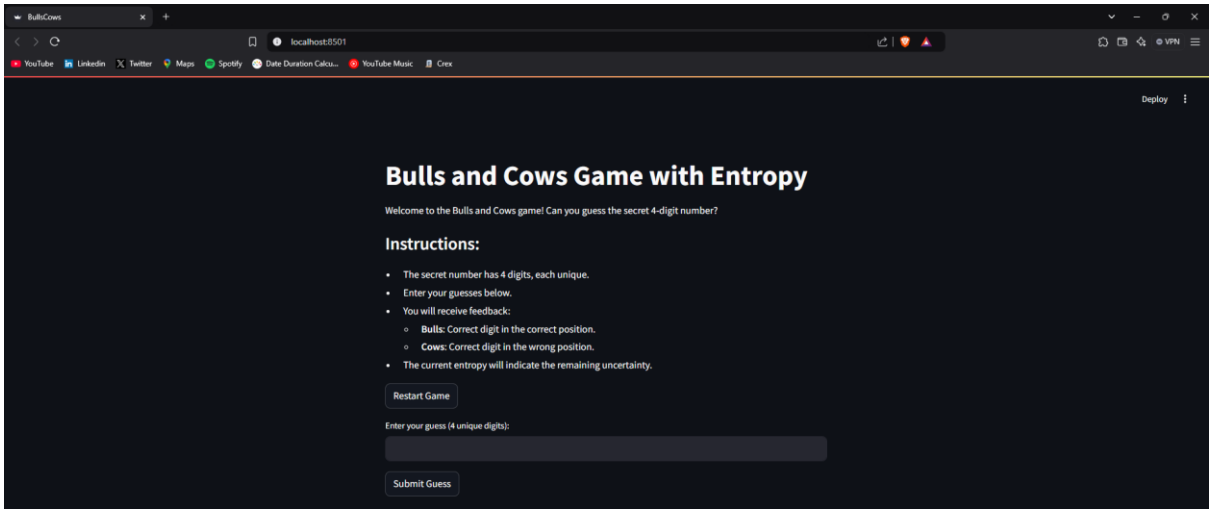
Secondly, the game logic includes **precise feedback** through the `evaluate_guess` function, which calculates the number of Bulls (correct digits in the correct position) and Cows (correct digits in the wrong position). This feedback drives the dynamic filtering of valid states using `filter_possible_states`, which ensures the game remains aligned with the feedback provided to the player. This logical narrowing down of possibilities adds a layer of strategic depth to the gameplay.

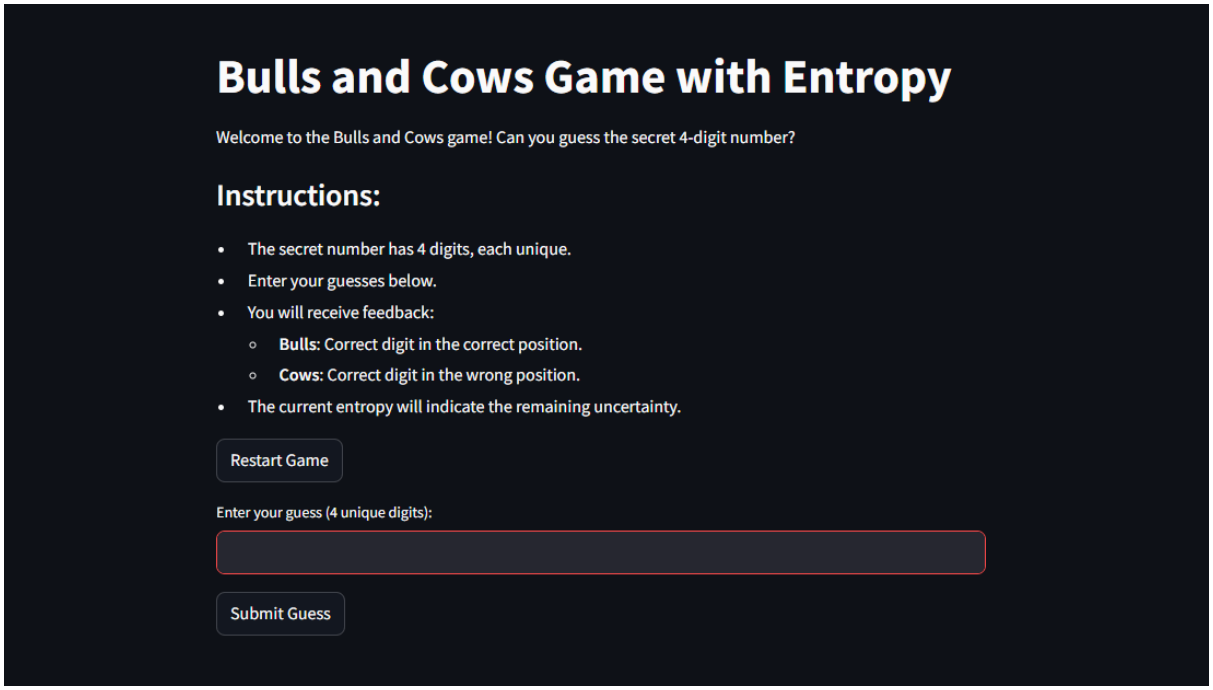
The third core functionality is **Entropy calculation**, implemented via the `calculate_entropy` function. Shannon entropy quantifies the uncertainty about the secret number based on the remaining valid states. As guesses eliminate invalid combinations, entropy reduces, providing players with a mathematical insight into their progress. Lastly, the user interface is interactive and responsive, allowing players to input guesses, view real-time feedback, and **track their progress through Entropy charts**. The interface also includes a "**Restart Game**" feature, enabling players to reset and start fresh anytime, ensuring accessibility and a continuous, engaging experience.

7. Sample Results

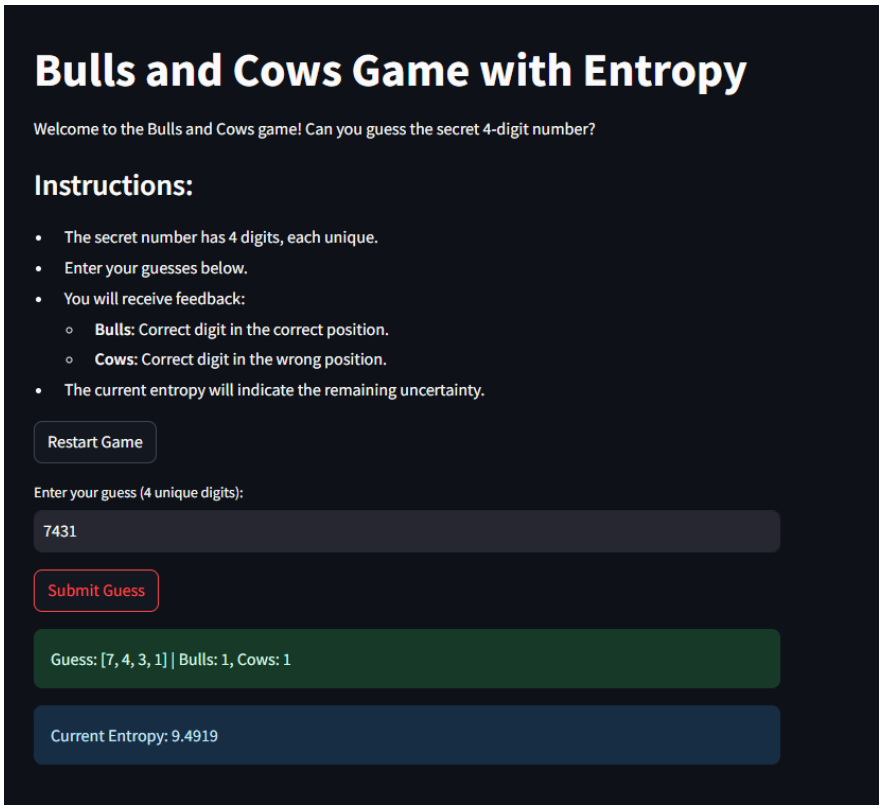
7.1 Sample Web Snapshots

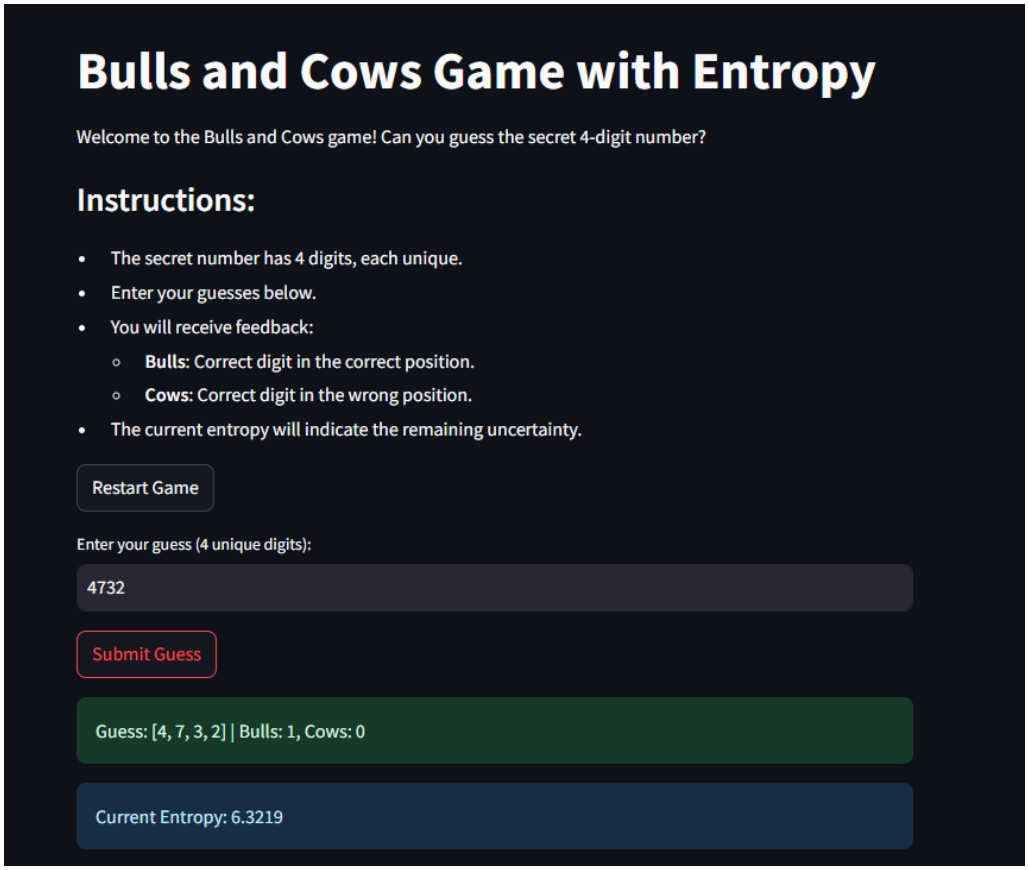
- **Initial State:**



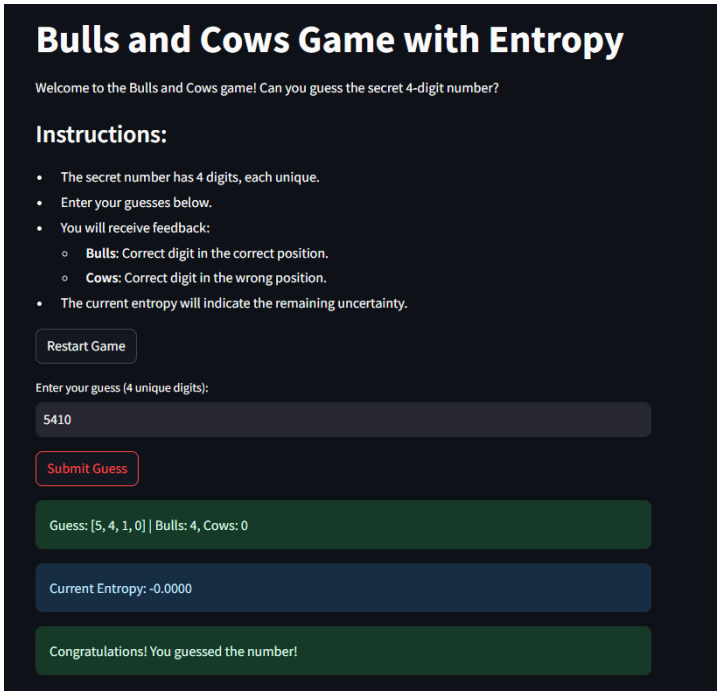


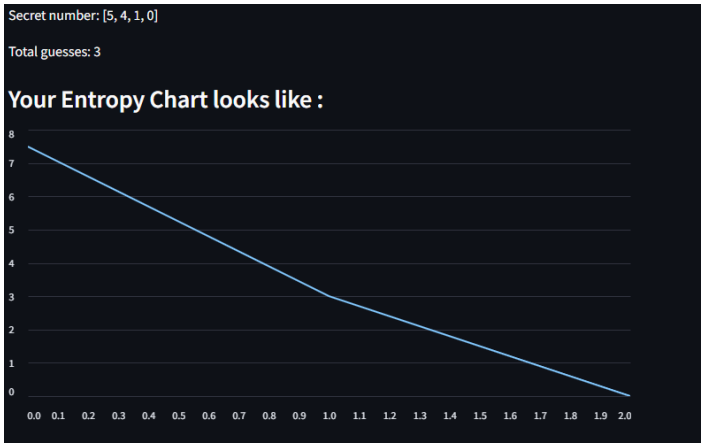
- **Mid-Game States:**

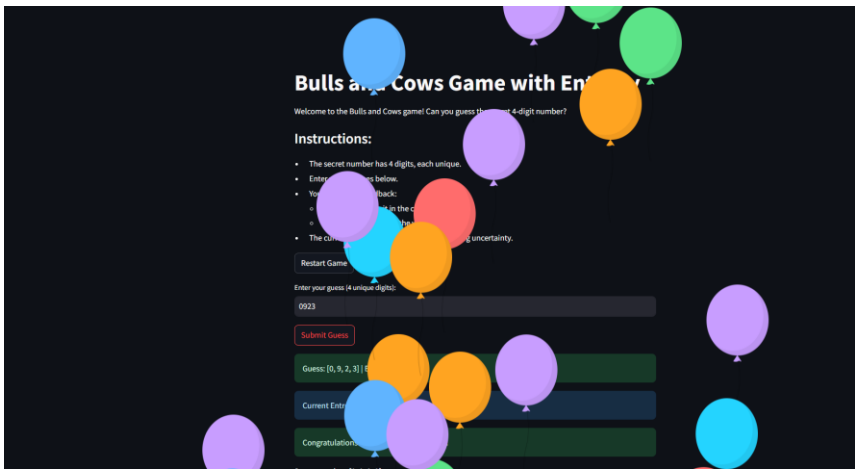




- **Final Winning State:**







8. CONCLUSION

The Bulls and Cows Game with Entropy project is an interesting integration of classic guessing game with advanced theoretical concepts from Information theory. This work bridges the gap between an engaging guessing game and the application of mathematical principles, delivering both an entertaining and educational experience for players.

At its core, this project demonstrates how feedback mechanisms systematically reduce uncertainty. By implementing Shannon entropy, we quantified the level of uncertainty at each stage of the game, providing players with insights into their progress. This dynamic reduction in entropy, coupled with the strategic feedback of Bulls and Cows, adds an analytical layer to the gameplay, making it both thought-provoking and enjoyable.

The project utilized Python as the primary programming language, showcasing its versatility in handling game logic, entropy calculations, and state management. It has Supporting libraries like `random`, `math`, and `itertools` efficiently handled the backend operations, including random number generation, combinatorial operations, and mathematical computations The use of libraries such as `Streamlit` enhanced the interactivity and accessibility of the application, allowing players to engage through a seamless web-based interface.

Additionally, the user interface designed with Streamlit played a significant role in making the game user-friendly and visually appealing. The interface included real-time feedback for guesses, dynamic entropy visualization through charts, and interactive controls for restarting the game. These features contributed to creating a highly interactive and engaging environment for players.

In conclusion, the **Bulls and Cows Game with Entropy** successfully met its objectives by blending practical game mechanics with theoretical insights. It highlighted the real-world application of entropy, engaged players in logical problem-solving, and showcased how computational tools can bring theoretical concepts to life.