Bulls and Cows Game with Entropy

Purpose of the Project

The purpose of this project is to explore and demonstrate the practical application of Shannon's entropy, a foundational concept in information theory, to the game 'Bulls and Cows.' Inspired by the practical challenges of decision-making under uncertainty, the project applies entropy to optimize gameplay by systematically reducing uncertainty with each step. By leveraging feedback loops, it highlights how theoretical constructs like entropy can simplify problem-solving, improve efficiency, and provide actionable insights into performance trends.

This project serves as an example of how theoretical concepts can directly translate into real-world applications. Beyond games, the principles demonstrated here are relevant to fields like cryptography, machine learning, and data compression, showcasing the universal importance of entropy in decision-making processes.

The accompanying presentation slides provide a detailed overview of the theoretical background, gameplay insights, and practical implications of entropy in 'Bulls and Cows.' These slides complement the video and documentation by offering a concise, visual summary of the project.

Setup Instructions

To get everything ready for this project, I started by ensuring all the necessary libraries were installed in my Python environment. The main libraries I used include:

- colorama for terminal text styling.
- Visualization libraries like matplotlib and seaborn.
- Data handling tools like numpy and pandas.

To make sure everything worked smoothly, I installed the required packages using pip:
pip install colorama matplotlib seaborn numpy pandas

Once the environment was set up, I tested it by importing these libraries in a small script to verify there were no installation issues. After that, I made sure all the code and any datasets were in the same directory so that everything was easy to access when running the project.

Usage Guide

For this project, I used the `GameTracker` class to track and analyze game progress. Here are some key methods:

```
# Class definition
class GameTracker:
    def __init__(self):
        pass
```

To log game statistics, I used the `add_game()` method:

```
# Adding game data
def add_game(self, game_stats):
    # Code to store game stats
    pass
```

For visualizing progress, the `visualize_progress()` method generates detailed graphs:

```
# Visualizing progress
def visualize_progress(self):
    # Code to generate graphs
    pass
```

Practical Implications

Machine Learning:

Entropy is used to calculate information gain, which helps in building efficient decision trees. This is a fundamental concept for improving the accuracy of predictive models.

Cryptography:

In secure communication, entropy ensures the strength of encryption keys. Higher randomness (or entropy) in the keys makes systems much harder to break.

Data Compression:

By identifying patterns in data, entropy helps reduce file sizes, which is crucial in technologies like ZIP files

and image compression algorithms.

Steps in Entropy Calculation

1. Make a Guess:

Start with a guess for the secret number. For example, in one instance, I guessed '1243.'

2. Receive Feedback:

Based on the secret number, I received feedback in terms of 'bulls' (correct digits in the correct positions) and 'cows' (correct digits but in the wrong positions). For the guess '1243,' if the secret number was '1234,' I might get 2 bulls and 2 cows.

3. Filter Possible Numbers:

Using the feedback, I eliminated all numbers that didn't fit the criteria (e.g., numbers that couldn't produce 2 bulls and 2 cows in response to the guess).

4. Calculate Entropy:

I then calculated the entropy of the remaining possibilities using Shannon's formula. Higher entropy indicated more uncertainty (more possible solutions), while lower entropy showed progress toward the correct answer.

5. Repeat the Process:

Each guess and feedback cycle reduced the entropy, shrinking the pool of possible solutions. When entropy reached zero, only one possible number remained - the correct answer.

```
# Entropy Calculation
def calculate_entropy(possible_numbers, guess=None, bulls=None, cows=None):
    # Code to calculate entropy
    pass
```

References and Resources

1. GitHub Repository:

Bulls-and-Cows-Game Repository

2. Videos:

Formal Presentation Video and Code Demonstration