



Entropy Exam
Mathematical Foundations for Data Science

Under the supervision
Laura Melissa Cruz Castro

Executed by:
YAGNESH CHALLAGUNDLA

TITLED: Bulls and Cows Game with Entropy

Link for the code (GitHub):

Link for the recording:

EXECUTED CODE:

```
 jupyter Maths Exam III [Information theory] Last Checkpoint: 2 hours ago (unsaved changes)  Logout
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
```

```
In [ ]: import tkinter as tk # I am importing tkinter as tk for the graphical user interface (GUI)
from tkinter import ttk, messagebox, StringVar, Listbox, Scrollbar # I am importing additional tkinter modules for widgets
import itertools # I am importing itertools to generate permutations of digits for possible secret codes
import math # I am importing math for mathematical operations like logarithm
import matplotlib.pyplot as plt # I am importing matplotlib's pyplot for plotting graphs
import random # I am importing random to randomly choose secret codes
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg # I am importing FigureCanvasTkAgg to embed matplotlib figure
import seaborn as sns # I am importing seaborn for enhanced statistical graphics

# Core Logic functions
def calculate_entropy(possible_codes): # I am defining the function to calculate entropy
    total_codes = len(possible_codes) # I am counting the total number of possible secret codes
    if total_codes == 0: # I am checking if there are no possible codes
        return 0 # I am returning 0 if there are no possible codes
    probability = 1 / total_codes # I am calculating the probability of each code (uniform distribution)
    entropy = -total_codes * probability * math.log2(probability) # I am applying the Shannon entropy formula
    return entropy # I am returning the calculated entropy

def bulls_and_cows(secret, guess): # I am defining the function to calculate bulls and cows for a guess
    bulls = sum(s == g for s, g in zip(secret, guess)) # I am counting the number of bulls (correct digits in the correct position)
    cows = sum(min(secret.count(d), guess.count(d)) for d in set(guess)) - bulls # I am counting the number of cows (correct digits in the wrong position)
    return bulls, cows # I am returning the bulls and cows counts

def filter_possible_codes(possible_codes, guess, feedback): # I am defining the function to filter possible codes based on feedback
    return [ # I am returning a list of codes that match the feedback
        code for code in possible_codes # I am iterating over each possible code
        if bulls_and_cows(code, guess) == feedback # I am filtering codes that match the bulls and cows feedback
    ]

def give_hint(secret_code, guessed_digits): # I am defining the function to give a hint
    unguessed = set(secret_code) - guessed_digits # I am finding the digits in the secret code that haven't been guessed
    if unguessed: # I am checking if there are still unguessed digits
        hint = random.choice(list(unguessed)) # I am randomly selecting a hint from the unguessed digits
        return hint # I am returning the hint
    else: # I am handling the case when there are no unguessed digits left
        return None # I am returning None if there are no unguessed digits

# GUI implementation
class BullsAndCowsGUI: # I am defining the GUI class for the Bulls and Cows game
    def __init__(self, root): # I am initializing the class with the root tkinter window
        self.root = root # I am storing the root window
        self.root.title("Bulls and Cows Game with Entropy") # I am setting the title of the root window
        self.root.geometry("1000x1000") # I am setting the size of the window to 1000x1000
        self.root.configure(bg="#3498db") # I am setting the background color of the window

        # Game variables
        self.secret_code = "" # I am initializing the secret code as an empty string
        self.possible_codes = [] # I am initializing an empty list to store possible codes
        self.attempts = 0 # I am initializing the number of attempts to 0
        self.previous_entropy = 0 # I am initializing the previous entropy value to 0
        self.entropy_history = [] # I am initializing an empty list to store the history of entropy values
        self.mutual_info_history = [] # I am initializing an empty list to store the history of mutual information values
        self.guessed_digits = set() # I am initializing an empty set to store guessed digits

        # UI Elements
        self.difficulty = StringVar(value="4") # I am initializing the difficulty level as a string variable
        self.guess_input = StringVar() # I am initializing a string variable for storing the user's guess
        self.feedback_label = None # I am initializing the feedback label to None
        self.entropy_label = None # I am initializing the entropy label to None
        self.mi_label = None # I am initializing the mutual information label to None
        self.guess_listbox = None # I am initializing the guess listbox to None
        self.scrollbar = None # I am initializing the scrollbar to None
```

```

self.setup_welcome_ui() # I am calling the method to set up the welcome UI
self.setup_game_ui() # I am calling the method to set up the game UI

def setup_welcome_ui(self): # I am defining the method to set up the welcome screen UI
    welcome_label = tk.Label(self.root, text="Welcome to Bulls and Cows with Entropy Game", # I am creating a welcome label
                             font=("Times New Roman", 16, "bold"), fg="white", bg="#3498db") # I am setting the font and background of the label
    welcome_label.pack(pady=5) # I am packing the welcome label with padding

    description_text = "Created for Entropy Exam (Mathematical Foundations for Data Science)" # I am defining the description text
    description_label = tk.Label(self.root, text=description_text, font=("Times New Roman", 12), # I am creating the description label
                                fg="white", bg="#3498db", justify="center") # I am setting the font and background of the label
    description_label.pack(pady=5) # I am packing the description label with padding

    name_label = tk.Label(self.root, text="Name of the Player:", font=("Times New Roman", 14, "bold"), fg="black", bg="#3498db") # I am creating a name label
    name_label.pack(pady=5) # I am packing the name label with padding

    self.name_entry = tk.Entry(self.root) # I am creating an entry widget for the player's name
    self.name_entry.pack(pady=5) # I am packing the name entry widget with padding

    start_button = tk.Button(self.root, text="Start Game", command=lambda: self.show_game_ui(), # I am creating a start button
                             bg="#2ecc71", fg="black", font=("Times New Roman", 12, "bold")) # I am setting the style of the button
    start_button.pack(pady=8) # I am packing the start game button with padding

    credits_button = tk.Button(self.root, text="Created by YAGNESH CHALLAGUNDLA", # I am creating a credits button
                               bg="#3498db", fg="white", font=("Arial", 10), borderwidth=0) # I am setting the style of the button
    credits_button.pack(pady=5) # I am packing the credits button with padding

def setup_game_ui(self):
    # Creating the main game frame with a specific background color
    self.game_frame = tk.Frame(self.root, bg="#3498db")

    # Adding a Label for difficulty selection
    tk.Label(self.game_frame, text="Select Difficulty:", bg="#3498db", fg="black").grid(row=0, column=0, pady=10, sticky="w")
    for i, difficulty in enumerate(["4 Digits", "5 Digits", "6 Digits"], start=1):
        tk.Button(self.game_frame, text=difficulty, command=lambda d=i+3: self.start_game(d)).grid(row=0, column=i)

```

```

    # Adding a Label and entry field for the user to input their guess
    tk.Label(self.game_frame, text="Enter Your Guess Number:", bg="#3498db", fg="black").grid(row=1, column=0, pady=10, sticky="w")
    tk.Entry(self.game_frame, textvariable=self.guess_input).grid(row=1, column=1)

    # Adding a button for submitting guesses
    tk.Button(self.game_frame, text="Submit Guess", command=self.make_guess).grid(row=1, column=2)

    # Creating a Label to display feedback for the user's guesses
    self.feedback_label = tk.Label(self.game_frame, text="Feedback will appear here.", fg="black", bg="#3498db")
    self.feedback_label.grid(row=2, column=0, columnspan=3, pady=10)

    # Adding a button for providing hints to the user
    tk.Button(self.game_frame, text="Give some Hint", command=self.give_hint_button).grid(row=3, column=0, columnspan=3)

    # Creating Labels to display current entropy and mutual information
    self.entropy_label = tk.Label(self.game_frame, text="Current Entropy: -", fg="black", bg="#3498db")
    self.entropy_label.grid(row=4, column=0, columnspan=3, pady=10)

    self.mi_label = tk.Label(self.game_frame, text="Mutual Information: -", fg="black", bg="#3498db")
    self.mi_label.grid(row=5, column=0, columnspan=3, pady=10)

    # Setting up a matplotlib figure and subplots for visualization
    self.figure = plt.Figure(figsize=(6, 2), dpi=90)
    self.ax1 = self.figure.add_subplot(121)
    self.ax2 = self.figure.add_subplot(122)
    self.canvas = FigureCanvasTkAgg(self.figure, master=self.game_frame)
    self.canvas.get_tk_widget().grid(row=6, column=0, columnspan=3)

    # Adding a Label for displaying previously entered guesses
    tk.Label(self.game_frame, text="Previously Entered Guesses:", bg="#3498db", fg="black").grid(row=7, column=0, pady=10)

    # Creating Listbox with Scrollbar
    self.guess_listbox = Listbox(self.game_frame, width=40, height=3)
    self.guess_listbox.grid(row=8, column=0, columnspan=3, pady=8)

```

```

    # Adding vertical scrollbar for the Listbox
    self.scrollbar = Scrollbar(self.game_frame, orient="vertical", command=self.guess_listbox.yview)
    self.scrollbar.grid(row=8, column=3, sticky="ns")
    self.guess_listbox.config(yscrollcommand=self.scrollbar.set)

    tk.Button(self.game_frame, text="Exit", command=self.root.quit).grid(row=9, column=0, columnspan=3, pady=10)

def show_game_ui(self):
    # Displaying the game rules in a popup and then show the game UI
    self.show_rules_popup()
    self.game_frame.pack(fill="both", expand=True)

def show_rules_popup(self):
    # Displaying the rules of the game in a popup window
    rules = (
        "Rules of Bulls and Cows with Entropy:\n\n"
        "1. The secret code is a unique number with no repeated digits will be assigned randomly.\n"
        "2. You need to guess the code in the least number of attempts.\n"
        "3. After each guess, you'll receive feedback:\n"
        "   --> Bulls: Correct digits in the correct positions.\n"
        "   --> Cows: Correct digits in the wrong positions.\n"
        "4. Use the feedback to refine your guesses and find the code.\n"
        "5. You can also request hints for help.\n\n"
        "Good luck!"
    )
    messagebox.showinfo("Game Rules", rules)

```

```

def start_game(self, code_length):
    # Initializing the game by generating a random secret code and setting up the game state
    all_codes = [''.join(code) for code in itertools.permutations('0123456789', code_length)]
    self.secret_code = random.choice(all_codes)
    self.possible_codes = all_codes[:]
    self.attempts = 0
    self.previous_entropy = calculate_entropy(self.possible_codes)
    self.entropy_history = [self.previous_entropy]
    self.mutual_info_history = []
    self.guessed_digits = set()

    # Updating the feedback label with the current game state
    self.feedback_label.config(text=f"Guess the secret {code_length}-digit code.")
    self.update_visualizations()

def make_guess(self):
    # Validating the user's guess and update the game state based on the feedback
    guess = self.guess_input.get()
    code_length = len(self.secret_code)

    # Validating the input
    if len(guess) != code_length or not guess.isdigit() or len(set(guess)) != code_length:
        messagebox.showerror("Invalid Input", f"Enter a {code_length}-digit unique number.")
        return

    # Incrementing attempts and calculate feedback (Bulls and Cows)
    self.attempts += 1
    bulls, cows = bulls_and_cows(self.secret_code, guess)
    self.feedback_label.config(text=f"Feedback: {bulls} Bulls, {cows} Cows")

    # Adding the guess to the listbox
    self.guess_listbox.insert("end", f"Guess {self.attempts}: {guess} - Bulls: {bulls}, Cows: {cows}")

    if bulls == code_length:
        messagebox.showinfo("Congratulations", f"You guessed the code in {self.attempts} attempts!")

```

```

        return

    # Updating the list of possible codes and entropy information
    feedback = (bulls, cows)
    self.possible_codes = filter_possible_codes(self.possible_codes, guess, feedback)

    current_entropy = calculate_entropy(self.possible_codes)
    mi = self.previous_entropy - current_entropy
    self.previous_entropy = current_entropy

    # Updating the entropy and mutual information labels
    self.entropy_history.append(current_entropy)
    self.mutual_info_history.append(mi)

    self.entropy_label.config(text=f"Current Entropy: {current_entropy:.2f}")
    self.mi_label.config(text=f"Mutual Information: {mi:.2f}")

    self.update_visualizations()

def update_visualizations(self):
    # Updating the matplotlib plots for entropy and mutual information
    self.ax1.clear()
    self.ax2.clear()

    self.ax1.plot(self.entropy_history, marker='o')
    self.ax1.set_title("Entropy Over Time")
    self.ax1.set_xlabel("Attempts")
    self.ax1.set_ylabel("Entropy")

    self.ax2.plot(self.mutual_info_history, marker='o', color='orange')
    self.ax2.set_title("Mutual Information")
    self.ax2.set_xlabel("Attempts")
    self.ax2.set_ylabel("Mutual Information")

    self.canvas.draw()

```

```

def give_hint_button(self):
    # Providing a hint by revealing one of the digits in the secret code
    hint = give_hint(self.secret_code, self.guessed_digits)
    if hint:
        self.guessed_digits.add(hint)
        messagebox.showinfo("Hint", f"One of the digits is: {hint}")
    else:
        messagebox.showinfo("Hint", "You have already guessed all digits!")

# Running the GUI Application
if __name__ == "__main__":
    root = tk.Tk()
    app = BullsAndCowsGUI(root)
    root.mainloop()

```

GUI OUTPUT:



Figure 1: Beginning of the page. It is a welcome page that accepts a username as input.

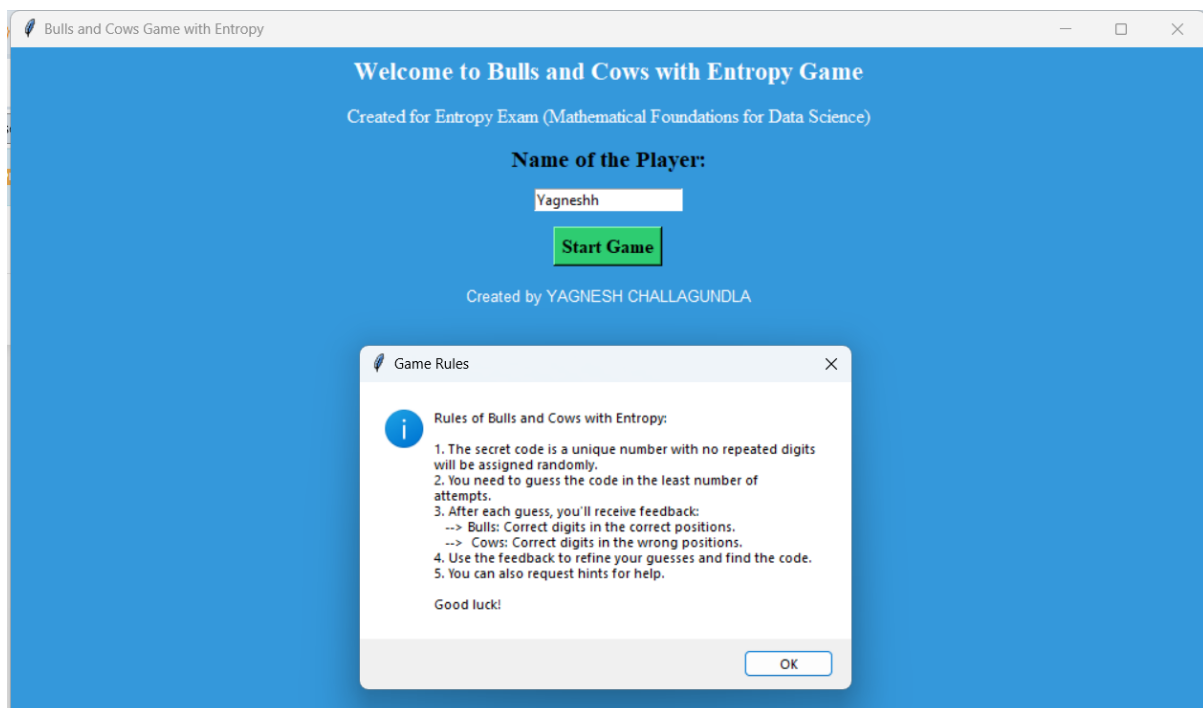


Figure 2: Upon entering the name, the game rules will be displayed.

Select Difficulty:

Enter Your Guess Number:

Feedback will appear here.

Figure 3: The difficulty level of the game must be chosen from 4 Digits, 5 Digits, or 6 Digits.

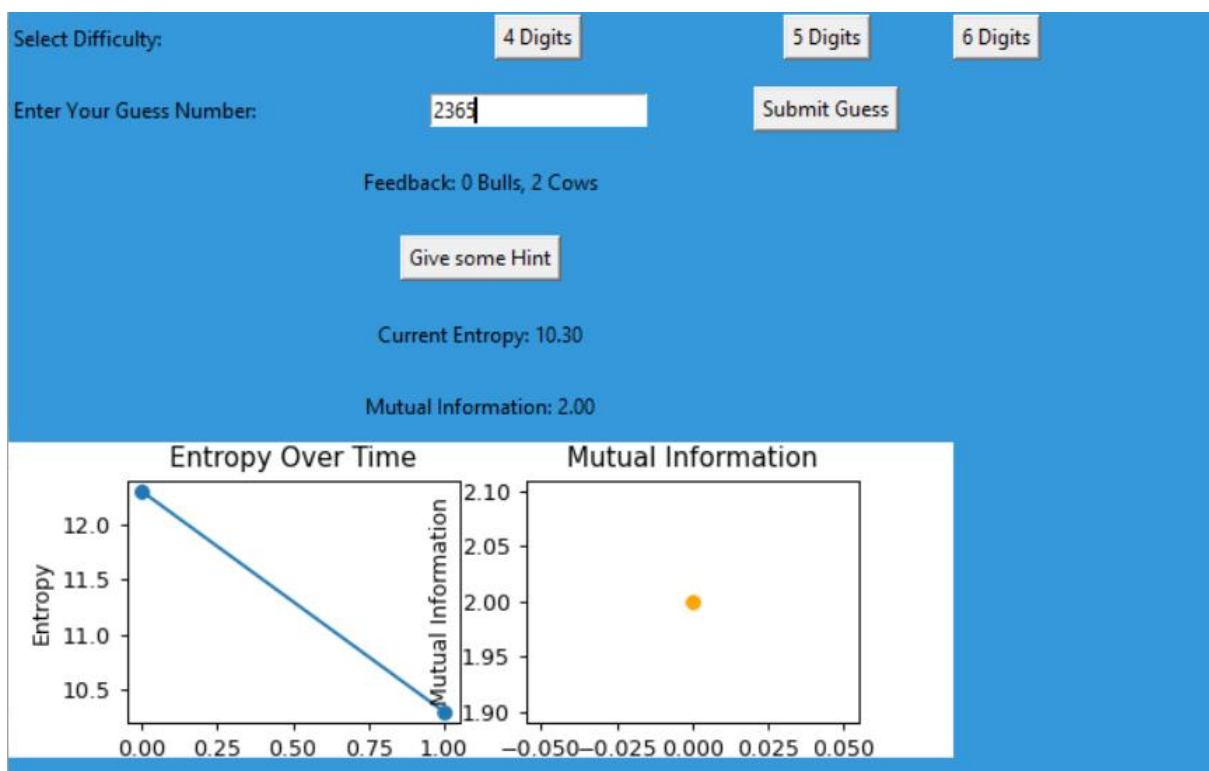


Figure 4: I selected four digits and provided an input of four distinct digits. The Bulls & Cows count, along with the current entropy value and mutual information value, is displayed, and a graph illustrating these metrics over time has also been included for enhanced visualization.

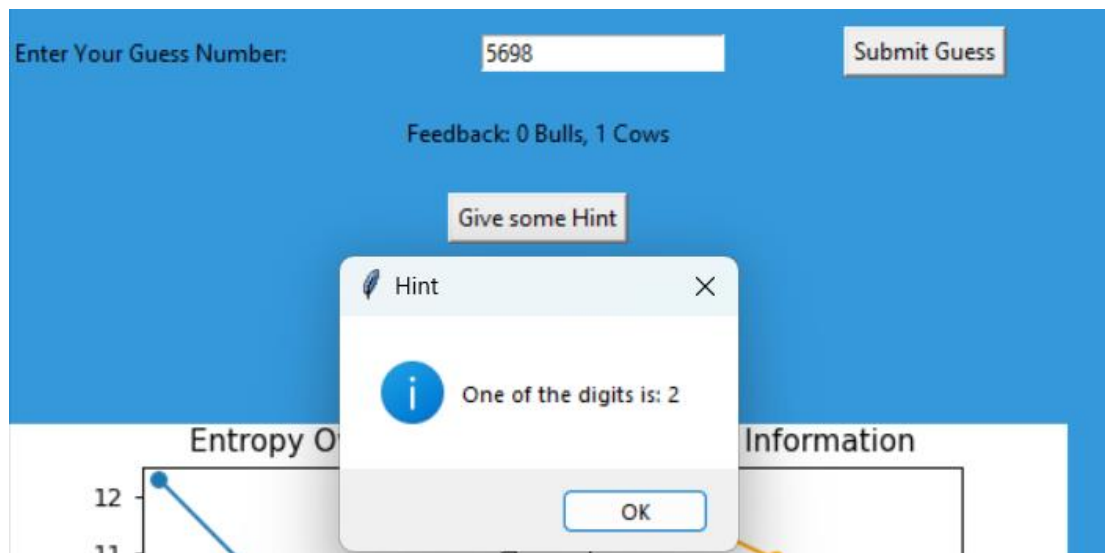


Figure 5: An option to click the hint button is presented, which reveals several available hints.

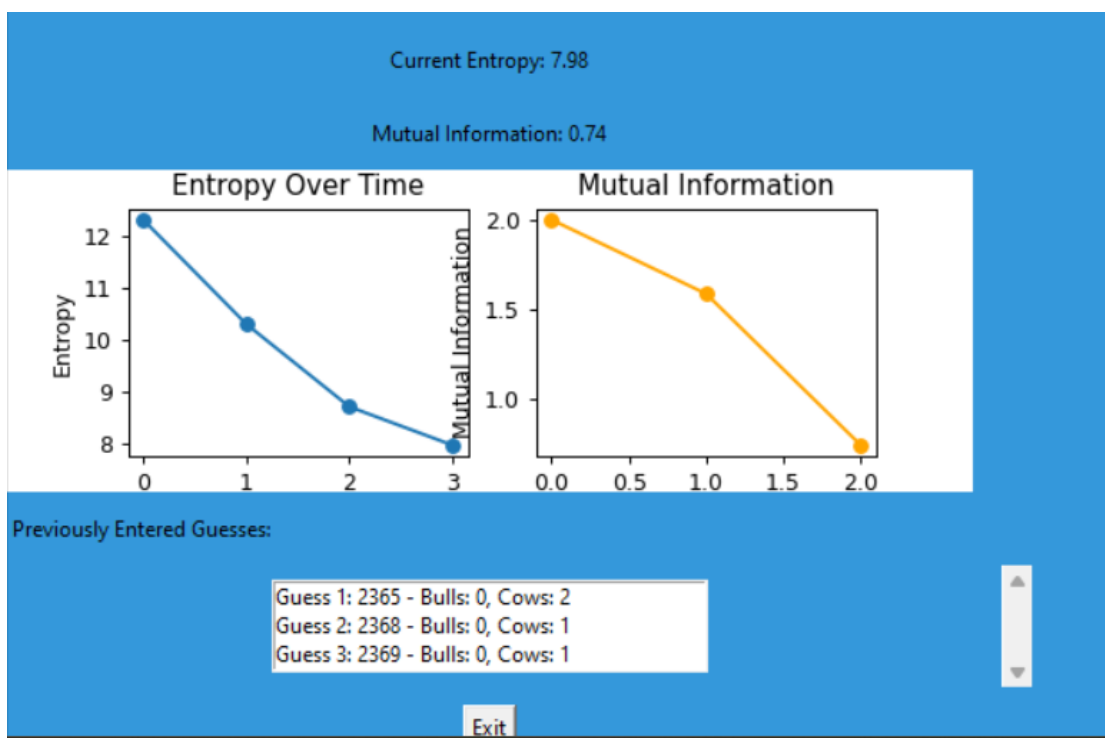


Figure 6: The graphs and previously entered values can be reviewed below, along with an option to exit the game mid-session.