Title Page Problem Statement: Implement a Rock-Paper-Scissors AI that predicts the player's move based on their history and counters it.

Name: Akshat
Roll No: 202401100400024
Date: 10-03-2025

Introduction The goal of this project is to create an AI that plays Rock-Paper-Scissors against a human player. The AI will analyze the player's past moves and attempt to predict their next move, choosing the best counter-move accordingly. This demonstrates basic machine learning principles and pattern recognition in a simple game environment.

Methodology

1. The AI tracks the player's move history.
2. It determines the most frequently played move and predicts that the player will repeat it.
3. The AI selects the counter-move accordingly.
4. The game runs in a loop, allowing multiple rounds with a scoring system.
5. The code is implemented in Python using Google Colab.

Code

```
import random

import matplotlib.pyplot as plt



def get_computer_move(history):

    """Returns the AI's move based on the player's history."""

    if not history:

        return random.choice(["rock", "paper", "scissors"])



    # Predict the most frequent move and counter it

    move_counts = {"rock": history.count("rock"), "paper": history.count("paper"), "scissors":
history.count("scissors")}
```

```
    prediction = max(move_counts, key=move_counts.get)


    # Counter the predicted move

    counter_moves = {"rock": "paper", "paper": "scissors", "scissors": "rock"}

    return counter_moves[prediction]


def determine_winner(player_move, computer_move):

    """Determines the winner of the game."""

    if player_move == computer_move:

        return "It's a tie!"


    winning_moves = {"rock": "scissors", "scissors": "paper", "paper": "rock"}


    if winning_moves[player_move] == computer_move:

        return "You win!"

    else:

        return "Computer wins!"


def plot_move_history(history):

    """Visualizes the player's move history using matplotlib."""

    if not history:

        print("No moves played. No graph to display.")

        return


    move_counts = {"rock": history.count("rock"), "paper": history.count("paper"), "scissors": 
history.count("scissors")}
```

```
    plt.figure(figsize=(6,4))

    plt.bar(move_counts.keys(), move_counts.values(), color=['red', 'blue', 'green'])

    plt.xlabel("Move Type")

    plt.ylabel("Frequency")

    plt.title("Final Player Move History")

    plt.ylim(0, max(move_counts.values(), default=1) + 1)  # Ensure a proper y-axis scale

    plt.show()


def play_game():

    """Main function to play Rock-Paper-Scissors with final visualization."""

    history = []

    score = {"player": 0, "computer": 0, "ties": 0}


    while True:

        player_move = input("Enter rock, paper, or scissors (or 'quit' to exit): ").lower()

        if player_move == "quit":

            print("Game Over!")

            print(f"Final Score - You: {score['player']}, Computer: {score['computer']}, Ties: {score['ties']}")

            plot_move_history(history)  # Display final move history graph

            break


        if player_move not in ["rock", "paper", "scissors"]:

            print("Invalid move. Please enter 'rock', 'paper', or 'scissors'.")

            continue
        computer_move = get_computer_move(history)

        print(f"Computer chose: {computer_move}")


        result = determine_winner(player_move, computer_move)

        print(result)


        # Update score

        if result == "You win!":

            score["player"] += 1

        elif result == "Computer wins!":

            score["computer"] += 1

        else:

            score["ties"] += 1


        history.append(player_move)


# Run the game

play_game()
```

Output/Result A screenshot of the game execution in Google Colab is attached below. The AI effectively predicts and counters the player's moves, making the game dynamic and engaging. The final score is displayed at the end of the session.

References/Credits

- Python Official Documentation
- Google Colab for running the interactive script
- Various online resources for AI game strategies