INSTITUTE OF PRINTING TECHNOLOGY
AND
GOVERNMENT POLYTECHNIC COLLEGE
SHORANUR

OPEN-ENDED PROJECT REPORT

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING LAB

COURSE CODE: 5139C

TOPIC :

STRATEGIC SHOWDOWN: AI-ENHANCED RPS DUEL

SUBMITTED BY:

SREENANDHAN P P - REG NO: 2101131235,ROLL NO: 46

SAHARSH KRISHNA C – REG NO: 21011311228,ROLL NO: 39

ARJUN CT – REG NO: 2101132343,ROLL NO: 56

SUBMITTED TO :

SAANI

LECTURER

DEPARTMENT OF COMPUTER ENGINEERING

IPT & GPTC SHORANUR

ABSTRACT

In the realm of classic games, Rock, Paper, Scissors stands as a timeless and universally recognized choice-based contest. In our project, "Smart RPS: AI-Powered Rock, Paper, Scissors Game," we introduce a novel twist to this age-old game by integrating artificial intelligence (AI) and predictive analytics.

The core objective of our project is to enhance the gaming experience by developing an AI opponent capable of predicting and responding to a player's moves strategically. This AI-driven RPS game leverages a history-based model that learns and adapts to the player's preferences, seeking to outsmart and challenge players with increasingly sophisticated strategies.

In addition to the advanced AI capabilities, our project retains the fundamental charm of RPS, enabling players to engage in strategic battles against the AI or with fellow human players. Players are invited to make their moves while the AI employs predictive algorithms to optimize its own choices.

This project serves as an engaging and educational demonstration of how AI can be integrated into traditional games to create dynamic and challenging experiences. It showcases the fusion of classic entertainment with cutting-edge technology, promising players an exhilarating and intellectually stimulating RPS showdown.

Join us on this journey to explore the world of strategic gaming with our "Smart RPS" project, where human wit meets machine intelligence, promising endless hours of strategic fun.

PROGRAM

```
import random

# Define the moves
moves = ["rock", "paper", "scissors"]

# Initialize a dictionary to store opponent's move history
opponent_moves = {
    "rock": 0,
    "paper": 0,
    "scissors": 0
}

# Define a matrix to store transition probabilities
transition_matrix = {
    "rock": {"rock": 0.33, "paper": 0.33, "scissors": 0.34},
    "paper": {"rock": 0.34, "paper": 0.33, "scissors": 0.33},
    "scissors": {"rock": 0.33, "paper": 0.34, "scissors": 0.33},
}

# Function to predict opponent's move and choose the winning move
def predict_and_play():
    expected_moves = {"rock": 0, "paper": 0, "scissors": 0}
    for move in moves:
        for opponent_move in moves:
            expected_moves[move] += opponent_moves[opponent_move] *
transition_matrix[opponent_move][move]


    bot_move = max(expected_moves, key=expected_moves.get)
    if bot_move == "rock":
        return "paper"
    elif bot_move == "paper":
        return "scissors"
    else:
        return "rock"

# Function to play a single game
def play_single_game(total_rounds):
    bot_score = 0
    human_score = 0

    for round_number in range(1, total_rounds + 1):
        bot_move = predict_and_play()

        human_move = input(f"Round {round_number}/{total_rounds} - Enter your move
(rock, paper, scissors, or 'exit' to quit): ").lower()

        if human_move == "exit":
            print("Exiting the game.")
            return

        if human_move not in moves:
            print("Invalid move. Please choose rock, paper, scissors, or 'exit' to quit.")
            continue

        opponent_moves[human_move] += 1

        if bot_move == human_move:
            print("It's a tie!")
        elif (
            (bot_move == "rock" and human_move == "scissors") or
```

```python
            (bot_move == "scissors" and human_move == "paper") or
            (bot_move == "paper" and human_move == "rock")
        ):
            print(f"Bot wins! Bot chose {bot_move}.")
            bot_score += 1
        else:
            print(f"You win! Bot chose {bot_move}.")
            human_score += 1

    print(f"Game Over - Final Score: Bot {bot_score}, Human {human_score}\n")

# Function to play the game
def play_game():
    print("Welcome to Rock, Paper, Scissors!")

    while True:
        total_rounds = 5  # Set the number of rounds per game

        play_single_game(total_rounds)

        play_again = input("Play another set of games? (yes/no): ").lower()
        if play_again != "yes":
            print("Thanks for playing!")
            return

if __name__ == "__main__":
    play_game()
```

RESULT: Succssfully builded AI bot to play RPS

OUTPUT:

```
Welcome to Rock, Paper, Scissors!
Round 1/5 - Enter your move (rock, paper, scissors, or 'exit' to quit): rock
Bot wins! Bot chose paper.
Round 2/5 - Enter your move (rock, paper, scissors, or 'exit' to quit): paper
You win! Bot chose rock.
Round 3/5 - Enter your move (rock, paper, scissors, or 'exit' to quit): scissors
You win! Bot chose paper.
Round 4/5 - Enter your move (rock, paper, scissors, or 'exit' to quit): rock
Bot wins! Bot chose paper.
Round 5/5 - Enter your move (rock, paper, scissors, or 'exit' to quit): paper
You win! Bot chose rock.
Game Over - Final Score: Bot 2, Human 3

Play another set of games? (yes/no): no
Thanks for playing!
```