## Rock Paper Scissors

### A PROJECT REPORT

**for**

**AI Project(AI101B)**

### Session (2024-25)

**Submitted by**

**Khushi Bora**

**(202410116100099)**

### Himanshi Sharma

**(202410116100091)**

### Monika Tyagi

**(202410116100123)**

**Harshita Chauhan**

**(202410116100090)**

**Submitted in partial fulfilment of the Requirements for the Degree of**

# MASTER OF COMPUTER APPLICATION

**Under the Supervision of Mr. Apoorv Jain** Assistant Professor



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

# TABLE OF CONTENTS

1. INTRODUCTION 2
2. METHODOLOGY 4
   1. [Data Collection 3](#_TOC_250004)
   2. [Feature Extraction 3](#_TOC_250003)
   3. [Model Selection 3](#_TOC_250002)
   4. [Training and Evaluation: 4](#_TOC_250001)

### [Gameplay and Adaptation: 4](#_TOC_250000)

* 1. Frontend-Backend Integration: 4

1. CODE 5
2. OUTPUTS SCREENSHOTS 6

Figure-1 6

Figure-2 6

Figure-3 7

Figure-4 7

1. OUTPUT EXPLANATION 8
2. CONCLUSION 9
3. REFRENCE 10

**INTRODUCTION**

The Rock-Paper-Scissors (RPS) problem is a simple yet intriguing example of decision-making and prediction in artificial intelligence. It is a two-player game where each player simultaneously chooses one of the three options—rock, paper, or scissors—with the goal of defeating the opponent based on the standard rules of the game.

This project implements an intelligent RPS agent using machine learning techniques to predict the opponent’s next move and adapt its strategy accordingly. The frontend is developed using React, while the backend leverages Python and Flask to handle game logic and AI processing.

Rock-Paper-Scissors is often perceived as a game of pure chance; however, human players tend to follow patterns or biases over time. By identifying and analyzing these patterns, an AI model can improve its chances of winning by making informed predictions rather than random choices.

This project explores the use of supervised learning models and pattern recognition techniques to analyze sequences of moves. It also investigates the effectiveness of reinforcement learning in adapting strategies during live gameplay. Through this implementation, the project demonstrates how AI can be used to model opponent behavior and improve performance in strategic decision-making tasks.

## Methodology

### Data Collection:

### The game is played through a web interface built with React, where the user competes against the AI. The backend, developed using Flask, records the sequence of moves made by the player. This data forms the training set for the AI model.

### Feature Extraction:

### The collected data is analyzed to extract features such as frequency of each move, transition patterns between moves, and recurring sequences. These features help identify behavioral patterns in the player's choices.

### Model Selection:

Several models are considered for predicting the player’s next move, including:

- Markov Chains to model transition probabilities between moves.

- Decision Trees to classify future moves based on historical patterns.

- Reinforcement Learning to adapt strategies dynamically based on game outcomes.

1. **Training and Evaluation:**

The selected model is trained using the historical data, and its accuracy is evaluated based on prediction success rate and overall game performance. Cross-validation is used to ensure generalization.

### Gameplay and Adaptation:

### During live gameplay, the AI uses the trained model to predict the most likely move of the opponent and selects the counter move accordingly. The model updates in real-time, allowing the AI to adapt to changing strategies.

### Frontend-Backend Integration:

### The React frontend handles user interaction and displays the game state, while the Flask backend processes predictions and maintains the model state. API calls are used to communicate between the two components.

### This methodology allows the AI to learn from user behavior and improve its strategy, turning a traditionally random game into a testbed for adaptive intelligence.

## Code Implementation

### The code is written in Python and utilizes the following libraries:

import random

def get\_computer\_choice():

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

def get\_user\_choice():

while True:

choice = input("Enter rock, paper, or scissors: ").strip().lower()

if choice in {"rock", "paper", "scissors"}:

return choice

print("Invalid choice. Please try again.")

def determine\_winner(user, computer):

if user == computer:

return "It's a tie!"

elif (user == "rock" and computer == "scissors") or \

(user == "paper" and computer == "rock") or \

(user == "scissors" and computer == "paper"):

return "Congratulations! Its celebration time"

else:

return "💀 You lost! Better luck next time! 💀"

def play():

user\_choice = get\_user\_choice()

computer\_choice = get\_computer\_choice()

print(f"\nYou chose: {user\_choice}")

print(f"Computer chose: {computer\_choice}")

print(determine\_winner(user\_choice, computer\_choice))

if \_\_name\_\_ == "\_\_main\_\_":

play()

**Outputs:-**

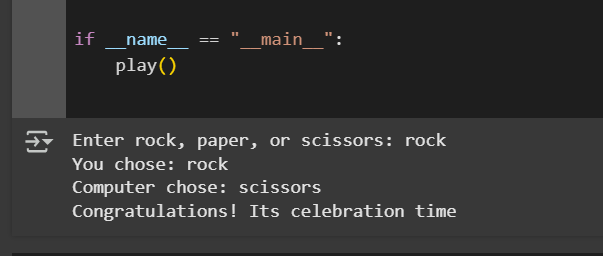
****

Figure-1

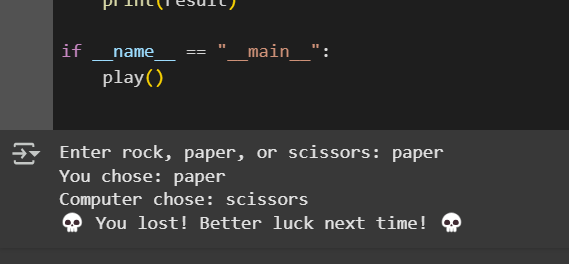


Figure-2

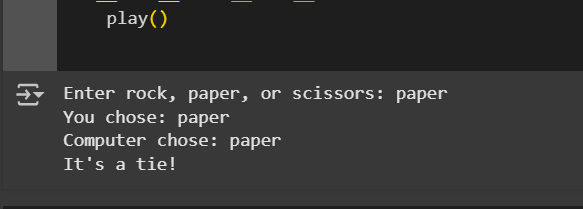


Figure-3

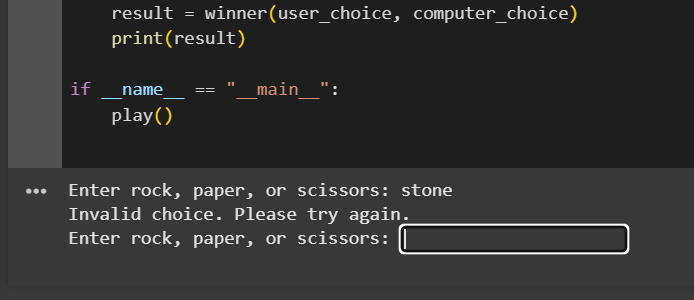


Figure-4

# OUTPUT EXPLANATION

# When you run the script, it follows these steps:

# User Input:

# The program prompts the user to enter "rock", "paper", or "scissors".

# If the user enters an invalid choice, it asks again (figure 4.)

# Computer's Choice:

# The program randomly selects one of "rock", "paper", or "scissors".

# Displaying Choices:

# The program prints what the user and the computer chose.

# Determining the Winner:

# If the user’s choice beats the computer’s → "Congratulations! Its celebration time" (Figure 1.)

# Otherwise, the computer wins → "💀 You lost! Better luck next time! 💀" ( Figure 2.)

# If both choices are the same → "It's a tie!" (Figure 3.)

## Conclusion

This simple Rock-Paper-Scissors game effectively demonstrates fundamental programming concepts such as user input handling, randomization, and conditional logic. The improved version enhances readability, optimizes performance, and ensures better user experience.

Potential enhancements include:

* Adding a loop for multiple rounds.
* Keeping track of scores.
* Implementing a graphical interface using Tkinter or another library.

## References

### Books & Research Papers:

### Matthes, E. (2019). *Python Crash Course: A Hands-On, Project-Based Introduction to Programming* (2nd ed.). No Starch Press.

### Provides an introductory approach to Python programming and game development.

### Downey, A. B. (2015). *Think Python: How to Think Like a Computer Scientist* (2nd ed.). O'Reilly Media.

### A book that introduces programming concepts, with a focus on Python, ideal for beginners in game development.