CENG 3511 Final Project Report

Ceyda Arık 210709014

Project Title

Rock-Paper-Scissors Game with Rule-Based and Machine Learning AI Agents

1. Introduction

This project implements an interactive Rock-Paper-Scissors (RPS) game where the user competes against an AI opponent. The game provides two AI options: a rule-based agent and a machine learning-based agent. The user can dynamically switch between these agents through the web-based interface built using Streamlit.

2. Game Definition

- **Game Type:** Turn-based, player vs. AI.
- **User Interaction:** The player selects rock, paper, or scissors via clickable buttons with corresponding emojis.
- **AI Role:** Acts as the opponent and plays based on either fixed rules or predictions learned from previous player moves.

3. AI Design

3.1 Rule-Based AI

- **Logic:** Selects a random move each round.
- **Purpose:** Simulates a fair, non-learning opponent.

3.2 Machine Learning AI

- **Method:** Decision Tree Classifier (using `scikit-learn`).
- **Input:** A one-hot encoded representation of previous player moves.
- **Prediction:** Predicts the player's next move and selects the winning counter-move accordingly.
- **Learning:** The model is retrained with new data after each round, starting after collecting 5 data points.

4. Implementation

4.1 Tools Used

- **Programming Language:** Python 3.12
- **Framework:** Streamlit for GUI and game logic
- **Libraries:** `scikit-learn`, `numpy`, `random`

4.2 Features

- Web-based GUI with emojis and clean layout.
- Real-time display of player and AI moves.
- Dynamic scoring system.
- AI mode selector in sidebar.
- Automatic reset and re-training upon mode change.

5. Experiments & Testing

- **Rule-Based AI:** Game behaves randomly and fairly.
- **ML-Based AI:** Initially behaves randomly, but starts adapting after 5 rounds by identifying repeating patterns in user moves.
- **Mode Switching:** Clean state reset ensures no data leakage between AI types, preserving fairness and experimental validity.

6. Results

- **User Feedback:** The interface is responsive, visually clear, and fun to use.
- **AI Behavior:** ML agent becomes slightly more competitive over time if the user has a predictable pattern.
- **Switching Logic:** Works correctly — resets model and score for fair comparison.

7. Challenges

- Ensuring AI doesn’t cheat by accessing the player's move before selecting its own.
- Managing session state and resetting cleanly when switching modes.

- Designing an intuitive GUI that works well for both gameplay and AI experimentation.

8. Conclusion

This project demonstrates how AI agents with different design philosophies (rule-based vs. ML) can interact in a simple game setting. It helped in understanding:

- How basic decision-making agents operate.
- How models learn patterns over time and use them for prediction.
- The role of UI/UX in making AI experiments accessible and engaging.

9. Future Work

- Add a reinforcement learning agent (Q-learning) for comparison.
- Store and visualize historical game data using charts.
- Improve prediction accuracy with advanced ML models like SVM or neural networks.

10. Repository & Setup

- **GitHub:** [Your GitHub Repo Link Here]
- **How to Run:**

```
pip install streamlit scikit-learn numpy
streamlit run rock_paper_ml.py
```