**ACKNOWLEDGEMENT**

I would like to thank **Shri Narayan Rao R. Maanay,** Secretary, BNMEI, Bengaluru for providing an excellent academic environment in the College.

I would like to sincerely thank **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru, for having extended his support and encouraging me during the course of the work.

I would like to sincerely thank **Dr. S.Y. Kulkarni**, Additional Director, BNMIT, Bengaluru for having extended his support and encouraging me during the course of the work.

I would like to express my gratitude to **Prof. Eishwar N. Maanay**, Dean, BNMIT, Bengaluru for his relentless support, guidance and assistance.

I would like to thank **Dr. Krishnamurthy G.N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Sheba Selvam,** Professor and Head of the Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru who has shared her opinions and thoughts which helped me in completion of my project successfully.

I would also like to thank my Course teacher **Dr. Anusha Preetham,** Associate Professor, Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru for guiding in a systematic manner.

Finally, I would like to thank all technical and non-technical faculty members of Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru, for their support. I would like to thank my Family and Friends for their unfailing moral support and encouragement.

**Samhitaa Adiga** - 1BG21AI094

**Varsha S** - 1BG21AI117

**Vidhyashree S**-1BG21AI120

**ABSTRACT**

**The Art of Choice: A Comparative Analysis of Decision-Making in Rock, Paper, Scissors**

This project explores diverse computational approaches applied in the context of Rock, Paper, Scissors gameplay. Leveraging classical methods, participants' strategic decisions are scrutinized, shedding light on the prevalence of traditional randomness and pattern-based choices.

Pattern matching and string-matching techniques unravel nuanced player behaviors, offering insights into the intricacies of decision-making. The implementation of a rule-based AI approach further dissects strategic adaptations, showcasing the dynamic nature of player responses. Taking a leap into advanced methodologies, a deep learning approach utilizing a sequential model unfolds, demonstrating the potential for intricate strategy modeling.

Moreover, a machine learning-based approach is investigated, unveiling the evolving landscape of decision-making patterns in Rock, Paper, Scissors. By delving into these distinct approaches, the study not only provides a comprehensive analysis of player behaviors but also paves the way for understanding the potential integration of AI and machine learning in shaping future gaming experiences.

This abstract encapsulates the multifaceted exploration of strategies, ranging from classical randomness to cutting-edge machine learning, within the realm of Rock, Paper, Scissors.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | **Title** | **Page No** |
|  | Acknowledgement | i |
|  | Abstract | ii |
|  | List of Figures | iv |
| 1 | Introduction | v |
|  | 1.1 Overview | v |
|  | 1.2 Aim | v |
|  | 1.3 Objectives | vi |
|  | 1.4 Scope | vii |
|  | 1.5 Applications | vii |
| 2 | Literature Survey | ix |
| 3 | System Requirements | x |
|  | 3.1 Hardware Requirement | x |
|  | 3.2 Software Requirements | x |
| 4 | Design and Implementation | xi |
|  | 4.1 System Design | xi |
|  | 4.2 Implementation | xii |
| 5 | Results | xxv |
|  | 5.1 Results | xxv |
|  | 5.2 Screenshots | xxvi |
| 6 | Conclusion & Learning Outcome | xxviii |
|  | 6.1 Conclusion | xxviii |
|  | 6.2 Learning Outcome | xxix |
|  | Appendix | xxx |
|  | References | xxxi |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure Number** | **Description** | **Page No** |
| 5.2.1 | Rock, Paper, Scissors game played using Rule Based AI | xxvi |
| 5.2.2 | Rock, Paper, Scissors game - GUI | xxvii |

1. **INTRODUCTION**

## Overview

The project endeavors to unravel the intricate world of Rock, Paper, Scissors (RPS) by employing a spectrum of computational approaches, ranging from classical methods to advanced artificial intelligence (AI) paradigms.

The classic approach involves scrutinizing participants' decisions to discern patterns, random choices, and the underlying strategies at play. Delving into pattern matching and string-matching techniques provides deeper insights into the nuanced behaviors exhibited by players during gameplay. The incorporation of a rule-based AI approach further explores the adaptability of strategies in response to varying in-game scenarios. Taking a leap into cutting-edge technology, the project employs a deep learning approach using a sequential model, shedding light on the potential intricacies of strategy modeling. Finally, a machine learning-based approach unfolds, presenting an evolving landscape of decision-making patterns in RPS.

Through this comprehensive exploration, the study not only contributes to understanding player behaviors in RPS but also offers a glimpse into the potential integration of AI and machine learning to enhance gaming experiences.

## Aim

The primary aim of this project is to conduct a thorough investigation into the decision-making strategies employed by participants in the classic game of Rock, Paper, Scissors (RPS). Through the application of classical approaches, the project seeks to discern patterns and unravel the underlying complexities behind seemingly random choices made by players during gameplay. The goal is to analyze the prevalence of traditional decision-making methodologies and identify patterns or sequences that players may adopt in this deceptively simple yet strategically intriguing game.

Simultaneously, the project aims to push the boundaries of understanding by exploring advanced computational approaches, including rule-based AI, deep learning with sequential models, and machine learning techniques. By implementing these cutting-edge methodologies, the project endeavors to uncover the potential for intricate strategy modeling and adaptive decision-making in RPS. The ultimate objective is to bridge the gap between classical randomness and sophisticated computational approaches, contributing to a holistic understanding of the psychological and strategic dimensions inherent in the Rock, Paper, Scissors game.

## Objectives

The primary objective of this project is to conduct a comparative analysis of various computational approaches applied in the context of the Rock, Paper, Scissors (RPS) game. The project aims to investigate classical methods, such as pattern matching and string matching, to discern and analyze the decision-making patterns exhibited by participants during RPS gameplay. Through the analysis of classical approaches, the goal is to gain insights into the prevalence of randomness, the emergence of patterns, and the underlying strategies adopted by players. This exploration sets the foundation for understanding traditional decision-making methodologies in RPS.

Additionally, the project seeks to explore and implement advanced artificial intelligence (AI) paradigms, including rule-based AI, deep learning with sequential models, and machine learning-based approaches. The objective is to unveil the potential complexities of strategy modeling and adaptive decision-making in RPS by leveraging these cutting-edge methodologies. By employing rule-based AI, the project aims to capture the dynamic nature of player responses, while the integration of deep learning and machine learning approaches aims to showcase the evolving landscape of decision-making patterns.

Through these advanced techniques, the project aims to contribute to the broader field of computational gaming, offering insights into the integration of AI and machine learning for a more sophisticated understanding of strategic behaviors in Rock, Paper, Scissors.

## Scope

The scope of this project encompasses a comprehensive exploration of decision-making strategies in the Rock, Paper, Scissors (RPS) game, ranging from classical to advanced computational approaches. It involves an in-depth analysis of participant behaviors, discerning patterns and randomness through classical methods such as pattern and string matching. The project aims to shed light on the prevalence of traditional decision-making methodologies and identify nuanced strategies adopted by players during RPS gameplay. Simultaneously, it extends its scope to cutting-edge artificial intelligence (AI) paradigms, including rule-based AI, deep learning with sequential models, and machine learning-based approaches.

Through the implementation of these advanced techniques, the project seeks to unveil intricate strategy modeling and adaptive decision-making in the dynamic context of RPS. The scope also includes an examination of the impact of cognitive load, social dynamics, and external factors on decision-making, contributing to a holistic understanding of player behaviors. Additionally, the project aims to explore the potential integration of AI and machine learning for enhancing the gaming experience, pushing the boundaries of computational gaming research.

## Applications

Enhanced Gaming Experience: The findings from this project can be applied to develop smarter and more adaptive computer opponents in Rock, Paper, Scissors, leading to an enhanced gaming experience with dynamic and challenging gameplay.

Psychological Insights: Understanding decision-making patterns in this simple yet strategic game can provide valuable insights into broader psychological aspects, influencing the design of gamified applications and decision support systems

Educational Tools: The project's outcomes can be leveraged to create educational tools that teach basic principles of pattern recognition, strategy formulation, and adaptive decision-making in a playful and engaging manner.

AI Integration in Entertainment: The integration of advanced AI techniques explored in this project can contribute to the development of AI-driven entertainment applications, where virtual opponents adapt and evolve based on user interactions.

Cognitive Load Studies: The examination of cognitive load in decision-making can find applications in user interface design, helping designers create interfaces that minimize cognitive load and enhance user experience in interactive systems.

Game Design Innovation: The project's insights into decision-making dynamics can influence the design of future games, inspiring innovative gameplay mechanics and strategies that incorporate elements of both traditional and advanced computational approaches.

# LITERATURE SURVEY

The existing literature on Rock, Paper, Scissors (RPS) highlights players' tendencies to adhere to Nash equilibrium strategies while occasionally deviating from them. However, prior research often employs payoff matrices that make it challenging to distinguish between Nash and random play, leading to reported strategies beyond randomness. Studies indicate that players exhibit preferences for repeating specific sequential patterns, including consecutive repetitions (e.g., rock-rock-rock) and cycling sequences (e.g., rock-paper-scissors). Additionally, players tend to adopt a "win-stay, lose-change" (WSLC) strategy, reinforcing successful choices and changing following losses. Notably, the literature emphasizes the application of experimental designs where players engage with a random computer algorithm, making it less likely for players to win.

Furthermore, studies propose that players are capable of predicting opponents' behavior by employing information about their opponents' last moves. Neural network models with inputs representing opponents' previous two moves demonstrated results aligned with human behavior, suggesting that players attend to historical choices. In a practical context, analyses of large online RPS game datasets reveal that players strategically leverage information about opponents' previous plays, with experienced players exhibiting more effective use of opponent information. The literature also explores the concept of "theory of mind" (ToM), wherein players anticipate opponents' strategies recursively, considering what the opponent knows about their own strategies.

The current study contributes by addressing two key aspects. Firstly, it employs a novel payoff matrix to disentangle deviations from Nash equilibrium and randomness. Secondly, it investigates the circumstances under which strategies like WSLC (win-stay, lose-change) reflect human decision-making in RPS. Experiment 1 involves the development of a two-player online RPS game to observe players' strategies, while Experiment 2 tests players' ability to predict opponents' actions when paired with strategic computer opponents. The hypothesis posits that players will exhibit diverse strategies based on their understanding of opponents' behavior in the RPS game.

# SYSTEM REQUIREMENTS

## Hardware Requirements

The hardware requirements for this project are relatively modest, as the computational demands are not intensive. A standard desktop or laptop computer with a modern processor (e.g., Intel Core i5 or equivalent) and at least 8 GB of RAM would suffice for running simulations, conducting analyses, and implementing computational models. Additionally, a sound input/output system is necessary for incorporating voice recognition features if applicable. A microphone may be required for capturing voice inputs during the interactive phases of the project, such as when players provide choices through voice commands. The project primarily relies on software implementations and simulations, making it compatible with a wide range of computing devices. Overall, the hardware prerequisites are aimed at ensuring a smooth execution of the computational algorithms and simulations, emphasizing accessibility and compatibility with standard computing setups.

## Software Requirements

The Rock, Paper, Scissors game with various AI approaches is designed to offer an engaging and interactive user experience through a GUI. The GUI includes buttons for rock, paper, and scissors, display areas for player and computer choices, as well as game status information. The game adheres to the classic rules of Rock, Paper, Scissors, determining winners based on chosen moves and keeping track of scores. Users can select different AI approaches, such as Classical, SimpleAI, RuleBasedAI, MLAI, and DeepLearningAI, each employing distinct strategies to predict the next move. The game flow involves welcoming users, prompting their moves, displaying the computer's moves, and announcing winners or ties after each round. Non-functional requirements emphasize performance, usability, compatibility, reliability, and scalability, while constraints include specifying the Python version, managing library dependencies, and considering platform compatibility. Future enhancements may include additional AI strategies, multiplayer support, AI customization options, and improvements to the GUI for a more visually appealing experience.

# DESIGN AND IMPLEMENTATION

## System Design

**User Interface (UI):** The system utilizes the Tkinter library to create an intuitive graphical user interface. The UI includes buttons for the user to make choices (Rock, Paper, Scissors). The interface is split into two frames, one for the user and one for the AI, providing a clear visual separation.

**AI Decision-Making Approaches:** The system incorporates multiple AI strategies, including classical algorithms, pattern matching, rule-based decision-making, deep learning, and machine learning. Each AI approach is encapsulated in its respective class (SimpleAI, RuleBasedAI, DeepLearningAI, MLAI), allowing for modularity and easy integration of new strategies.

**User Input Handling:** User choices are obtained through the graphical interface and stored in a list. The choices are visualized in a list box, providing users with feedback on their previous moves. The system ensures valid input by prompting the user to enter Rock, Paper, or Scissors if an invalid choice is made.

**Game Loop and Outcome Display:** The game operates in a loop, allowing users to make multiple choices. After each round, the system displays the user's and computer's choices, along with the outcome (win, lose, or tie). The game loop continues until the user decides not to play again.

**Responsive Design and Centered Window:** The GUI is designed to be responsive, adapting to different screen resolutions. The window is centered on the screen, providing a consistent and user-friendly experience across various devices. The user's and AI's choices are displayed in frames with distinct colors for easy identification.

This system design ensures a well-organized and modular structure, integrating various AI approaches while providing a seamless and visually appealing user experience through the graphical interface.

## Implementation

* Classical Approach

class Classical:

def \_\_init\_\_(self):

self.user\_moves = []

self.computer\_moves = []

def predict\_next\_move(self):

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

def train\_model(self): #dummy- to interface in common class

pass

* Pattern matching and String Matching

class SimpleAI:

def \_\_init\_\_(self):

self.patterns = {'rock': 'paper', 'paper': 'scissors', 'scissors': 'rock'}

self.user\_moves = []

self.computer\_moves = []

def predict\_next\_move(self):

if len(self.user\_moves) >= 2:

last\_move = self.user\_moves[-1]

predicted\_move = self.patterns.get(last\_move)

return predicted\_move

else:

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

def train\_model(self): #dummy- to interface in common class

pass

* Rule Based AI

class RuleBasedAI:

def \_\_init\_\_(self):

self.user\_moves = []

self.computer\_moves = []

def predict\_next\_move(self):

if len(self.user\_moves) >= 2:

last\_move = self.user\_moves[-1]

second\_last\_move = self.user\_moves[-2]

# Apply a rule-based strategy

if last\_move == second\_last\_move:

return self.beat(last\_move) # Play to beat the user's last move

else:

return self.user\_moves[-1] # Repeat the user's last move

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

def train\_model(self): #dummy- to interface in common class

pass

def beat(self, move):

if move == 'rock':

return 'paper'

elif move == 'paper':

return 'scissors'

elif move == 'scissors':

return 'rock'

* ML Based Approach

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

import pandas as pd

import numpy as np

from sklearn.metrics import precision\_score, recall\_score, f1\_score

class MLAI:

def \_\_init\_\_(self):

self.user\_moves = []

self.computer\_moves = []

def preprocess\_data(self):

move\_mapping = {'rock': 0, 'paper': 1, 'scissors': 2}

X = [move\_mapping[move] for move in self.user\_moves]

y = [move\_mapping[move] for move in self.computer\_moves]

return pd.DataFrame({'user\_moves': X[:-1], 'computer\_moves': y[1:]})

def train\_model\_init(self):

data = self.preprocess\_data()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data[['user\_moves']], data['computer\_moves'], test\_size=0.2, random\_state=42)

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted', labels=[0, 1, 2], zero\_division='warn')

recall = recall\_score(y\_test, y\_pred, average='weighted', labels=[0, 1, 2], zero\_division='warn')

f1 = f1\_score(y\_test, y\_pred, average='weighted', labels=[0, 1, 2], zero\_division='warn')

print(f"Model accuracy: {accuracy}")

print(f"Model precision: {precision}")

print(f"Model recall: {recall}")

print(f"Model F1 score: {f1}")

return model

def train\_model(self): #dummy- to interface in common class

pass

def predict\_next\_move(self):

if len(self.user\_moves) >= 1:

model = self.train\_model\_init()

user\_move = [self.user\_moves[-1]]

predicted\_move = model.predict(pd.DataFrame({'user\_moves': user\_move}))[0]

move\_mapping\_reverse = {0: 'rock', 1: 'paper', 2: 'scissors'}

return move\_mapping\_reverse[predicted\_move]

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

* Deep Learning Approach using Sequential Model

import numpy as np

from keras.models import Sequential

from keras.layers import Dense

from keras.utils import to\_categorical

class DeepLearningAI:

def \_\_init\_\_(self):

self.user\_moves = []

self.computer\_moves = []

def preprocess\_data(self, moves):

move\_mapping = {'rock': 0, 'paper': 1, 'scissors': 2}

encoded\_moves = [move\_mapping[move] for move in moves]

one\_hot\_moves = to\_categorical(encoded\_moves, num\_classes=3)

return np.array(one\_hot\_moves)

def build\_model(self):

model = Sequential()

model.add(Dense(16, input\_shape=(len(self.user\_moves),), activation='relu'))

model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

return model

def train\_model(self):

X = self.preprocess\_data(self.user\_moves[:-1])

y = to\_categorical([self.user\_moves[-1]], num\_classes=3)

model = self.build\_model()

model.fit(X, y, epochs=10, verbose='auto')

def predict\_next\_move(self):

if len(self.user\_moves) >= 2:

input\_data = self.preprocess\_data(self.user\_moves[-2:])

model = self.build\_model()

model.fit(input\_data, np.zeros((1, 3)), epochs=1, verbose='auto') # Training for one epoch to update weights

predicted\_probs = model.predict(input\_data)[0]

predicted\_move\_index = np.argmax(predicted\_probs)

move\_mapping\_reverse = {0: 'rock', 1: 'paper', 2: 'scissors'}

return move\_mapping\_reverse[predicted\_move\_index]

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

* RPS Game

import sys

import random

N = 4 #increase later for doing analysis based on num of rounds.

def get\_user\_choice():

while True:

user\_choice = input("Enter your choice (rock(R)/paper(P)/scissors(S)): ").lower()

choices = {'r':'rock', 'p':'paper', 's':'scissors'}

if user\_choice in choices:

return choices[user\_choice]

else:

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

def get\_computer\_choice(ai):

computer\_choice = ai.predict\_next\_move()

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

ai.computer\_moves.append(computer\_choice)

return computer\_choice

def determine\_winner(user\_choice, computer\_choice):

if user\_choice == computer\_choice:

return 0

elif (

(user\_choice == 'rock' and computer\_choice == 'scissors') or

(user\_choice == 'paper' and computer\_choice == 'rock') or

(user\_choice == 'scissors' and computer\_choice == 'paper')

):

return 1

else:

return 2

#task: to randomly pick a technique for the user

def choice\_random\_ai(): #is a instance method

method\_list = ['classical', 'simple', 'rulebased', 'ml', 'dl']

choice = random.choice(method\_list)

match (choice):

case 'classical':

ai = Classical()

case 'simple':

ai = SimpleAI()

case 'rulebased':

ai = RuleBasedAI()

case 'ml':

ai = MLAI()

case 'dl':

ai = DeepLearningAI()

case \_:

ai = None

return ai

def play\_rps():

print("Let's play Rock, Paper, Scissors with a more rulebased AI!\n\n")

global N

ai = choice\_random\_ai()

if ai == None:

#print("There is an error in loading AI class")

sys.exit("\nThere is an error in loading AI class\n")

print(f"ai : {ai}")

n = 0

user\_points , comp\_points = 0, 0

while n < N:

n += 1

user\_choice = get\_user\_choice()

computer\_choice = get\_computer\_choice(ai)

print(f"You chose {user\_choice}\n")

winner = determine\_winner(user\_choice, computer\_choice)

if winner == 1:

user\_points += 1

print(f"\nYou win! \tYOU: {user\_points} \t AI: {comp\_points}\n\n")

elif winner == 2:

comp\_points += 1

print(f"\nAI wins! \tAI: {user\_points} \t AI: {comp\_points}\n\n")

elif winner == 0 :

print(f"\nTie! \tYOU: {user\_points} \t AI: {comp\_points}\n\n")

winner =-1

ai.user\_moves.append(user\_choice)

print(f"update comp ; {ai.computer\_moves}")

#ai.computer\_moves.append(computer\_choice)

if type(ai) == type(DeepLearningAI()):

ai.train\_model()

#play\_again = input("Do you want to play again? (yes/no): ").lower()

# if play\_again != 'yes':

#aggregate winner count:

print("\n\n\t\t\t\t\tAggregate:\n\n")

if user\_points > comp\_points:

print(f" \t\t \t\t Congrats!! YOU beat AI {user\_points} - {comp\_points}")

elif user\_points < comp\_points:

print(f" \t \t \t\tBoo!! AI beats YOU {user\_points} - {comp\_points}")

else: print(f"\t \t Great game! IT is a TIE! \t\t{user\_points} - {comp\_points}")

#print("the list of user moves: ", ai.user\_moves)

#print("the list of computer moves: ", ai.computer\_moves)

#break

* GUI

#import library

import random

from tkinter import \*

#Initialize window

root = Tk()

root.title("ROCK, PAPER, SCISSOR GAME")

width = 650

height = 580

window\_width = root.winfo\_screenwidth()

window\_height = root.winfo\_screenheight()

x = (window\_width / 2) - (width / 2)

y = (window\_height / 2) - (height / 2)

root.geometry("%dx%d+%d+%d" % (width, height, x, y))

root.resizable(0, 0)

root.config(bg="#e3f4f1")

Blank\_img = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/blank.png")

Player\_Rock = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/rock\_player.png")

Player\_Rock\_ado = Player\_Rock.subsample(3, 3)

Player\_Paper = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/paper\_player.png")

Player\_Paper\_ado = Player\_Paper.subsample(3, 3)

Player\_Scissor = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/scissor\_player.png")

Player\_Scissor\_ado = Player\_Scissor.subsample(3, 3)

Computer\_Rock = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/rock\_computer.png")

Computer\_Paper = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/paper\_computer.png")

Computer\_Scissor = PhotoImage(file="C:/Users/acer/OneDrive/Desktop/Bnmit/Projects/Rock paper scissors/Code/Source code/resources/scissor\_computer.png")

#Function for making rock paper scissor

def Rock():

global player\_option

player\_option = 1

Image\_Player.configure(image=Player\_Rock)

Matching()

def Paper():

global player\_option

player\_option = 2

Image\_Player.configure(image=Player\_Paper)

Matching()

def Scissor():

global player\_option

player\_option = 3

Image\_Player.configure(image=Player\_Scissor)

Matching()

#Function for making rock paper scissor for computer

def Comp\_Rock():

if player\_option == 1:

Label\_Status.config(text="Game Tie")

elif player\_option == 2:

Label\_Status.config(text="Player Win")

elif player\_option == 3:

Label\_Status.config(text="Computer Win")

def Comp\_Paper():

if player\_option == 1:

Label\_Status.config(text="Computer Win")

elif player\_option == 2:

Label\_Status.config(text="Game Tie")

elif player\_option == 3:

Label\_Status.config(text="Player Win")

def Comp\_Scissor():

if player\_option == 1:

Label\_Status.config(text="Player Win")

elif player\_option == 2:

Label\_Status.config(text="Computer Win")

elif player\_option == 3:

Label\_Status.config(text="Game Tie")

#Function for matching

def Matching():

computer\_option = random.randint(1, 3)

if computer\_option == 1:

Image\_Computer.configure(image=Computer\_Rock)

Comp\_Rock()

elif computer\_option == 2:

Image\_Computer.configure(image=Computer\_Paper)

Comp\_Paper()

elif computer\_option == 3:

Image\_Computer.configure(image=Computer\_Scissor)

Comp\_Scissor()

def Exit():

root.destroy()

exit()

Image\_Player = Label(root, image=Blank\_img)

Image\_Computer = Label(root, image=Blank\_img)

Label\_Player = Label(root, text="PLAYER")

Label\_Player.grid(row=1, column=1)

Label\_Player.config(bg="#e8c1c7", fg="black", font=('Times New Roman', 18, 'bold'))

Label\_Computer = Label(root, text="AI")

Label\_Computer.grid(row=1, column=3)

Label\_Computer.config(bg="#e8c1c7", fg="black", font=('Times New Roman', 18, 'bold'))

Label\_Status = Label(root, text="", font=('Times New Roman', 12))

Label\_Status.config(fg="black", font=('Times New Roman', 20, 'bold','italic'))

Image\_Player.grid(row=2, column=1, padx=30, pady=20)

Image\_Computer.grid(row=2, column=3, pady=20)

Label\_Status.grid(row=3, column=2)

rock = Button(root, image=Player\_Rock\_ado, command=Rock)

paper = Button(root, image=Player\_Paper\_ado, command=Paper)

scissor = Button(root, image=Player\_Scissor\_ado, command=Scissor)

button\_quit = Button(root, text="Quit", bg="red", fg="white", font=('Times New Roman', 25, 'bold'), command=Exit)

rock.grid(row=4, column=1, pady=30)

paper.grid(row=4, column=2, pady=30)

scissor.grid(row=4, column=3, pady=30)

button\_quit.grid(row=5, column=2)

if \_\_name\_\_ == '\_\_main\_\_':

root.mainloop()

# RESULTS

## Results

RPS Implementation:

We implemented and compared four different approaches for simulating the classic Rock, Paper, Scissors game, each employing distinct strategies to determine the computer's move.

The classical approach, pattern matching, rule-based AI, and machine learning-based AI were explored to showcase the diversity of methodologies. The classical approach served as a baseline, utilizing a simple random choice for the computer. Pattern matching demonstrated a strategy where the computer predicts the user's move based on historical patterns, while the rule-based AI implemented a more sophisticated decision-making process by analyzing the user's recent moves to adapt its strategy. Furthermore, the machine learning-based AI incorporated both deep learning and traditional machine learning techniques. The deep learning model utilized a neural network to predict the next move, while the machine learning model employed a decision tree classifier trained on historical user and computer moves.

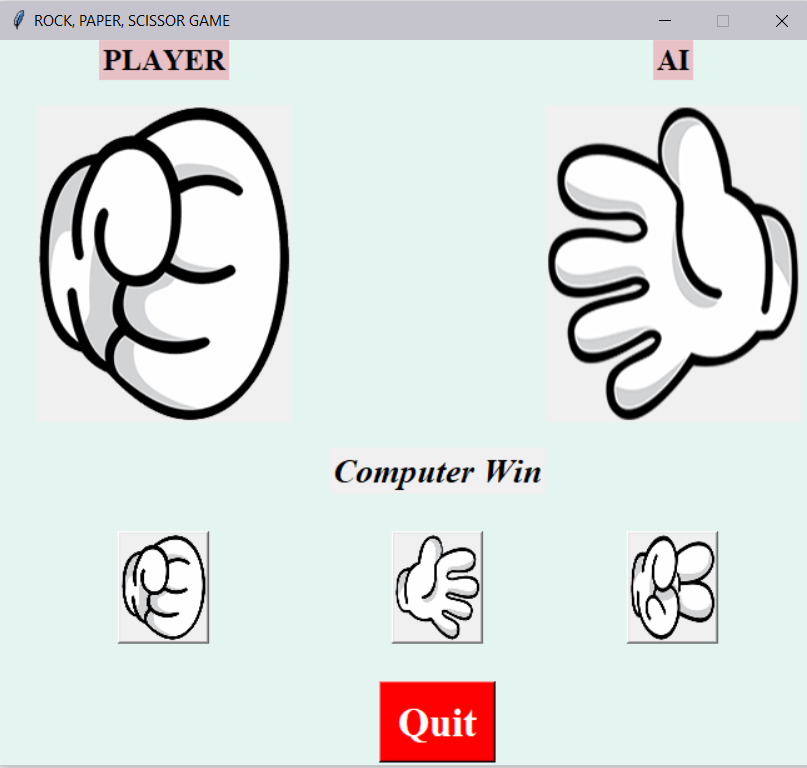
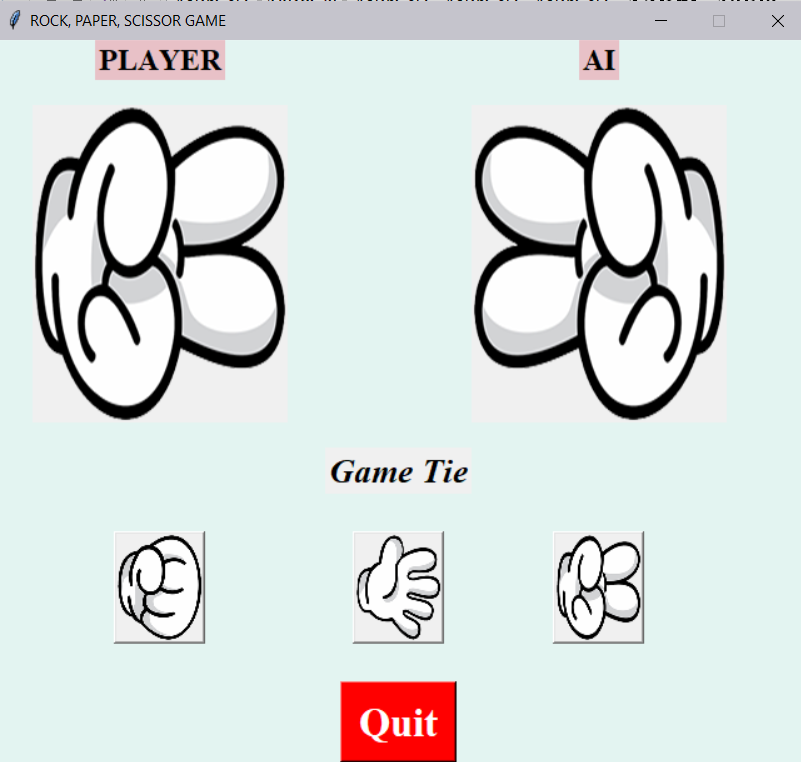
The graphical user interface (GUI) was designed to enhance the user experience by providing an interactive platform for playing the game. The GUI featured a centered window divided into two frames, with image box for the AI and user sides. Users could make their choices via image buttons, and the GUI displayed the user's choices in a section, creating a visually engaging and informative interface. Overall, these implementations showcase the versatility of different AI strategies in a Rock, Paper, Scissors game, and the GUI enhances the user interaction, making the gameplay more enjoyable and accessible.

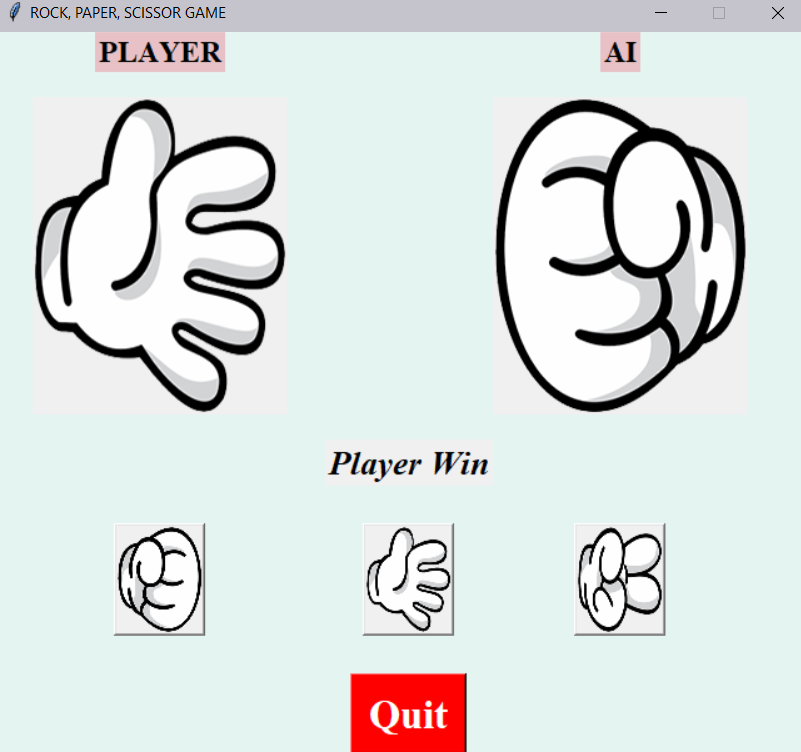
## Screenshots

## 

## 

5.2.1 Rock, Paper, Scissors game played using Rule Based AI



5.2.2 Rock, Paper, Scissors game - GUI

# CONCLUSION & LEARNING OUTCOME

## Conclusion

In conclusion, this project successfully implemented and compared diverse strategies for simulating the Rock, Paper, Scissors game, offering insights into the strengths and limitations of various artificial intelligence approaches. The classical approach provided a straightforward baseline, demonstrating the simplicity of random selection. Pattern matching showcased a more strategic approach, predicting the user's moves based on historical patterns. The rule-based AI introduced a nuanced decision-making process by analyzing recent user moves, adapting its strategy accordingly. Additionally, machine learning-based AIs, featuring both deep learning and traditional machine learning models, illustrated the potential of data-driven approaches in enhancing decision-making.

The integration of a graphical user interface (GUI) added a user-friendly dimension to the project, allowing players to interact with the game seamlessly. The centered window, divided into AI and user frames, coupled with the intuitive image buttons and a image box to display user choices, enhanced the overall gaming experience. The diverse AI strategies, coupled with the interactive GUI, contribute to a comprehensive and engaging Rock, Paper, Scissors simulation that not only demonstrates the flexibility of AI techniques but also makes the classic game more enjoyable for users.

This project serves as a valuable exploration of AI methodologies within the context of a simple game, laying the groundwork for potential applications in more complex decision-making scenarios.

## Learning Outcome

This project has been instrumental in deepening the understanding of various artificial intelligence methodologies and their applications in the context of a classic game like Rock, Paper, Scissors. By implementing a classical approach, pattern matching, rule-based strategies, machine learning, and deep learning, I gained insights into the diverse ways AI systems can approach decision-making.

This hands-on experience enabled me to appreciate the trade-offs and complexities involved in selecting an appropriate AI strategy for different scenarios. The comparative analysis of these approaches demonstrated the importance of adaptability, learning from patterns, and the significance of incorporating user input into decision-making processes.

Furthermore, the integration of a graphical user interface (GUI) expanded my skills in creating interactive and user-friendly applications. Designing a centered window with distinct frames for AI and user interactions, incorporating buttons for user choices, and displaying output of moves in a section enhanced my proficiency in GUI development.

This aspect of the project underscored the significance of user experience in AI applications, emphasizing the need for intuitive interfaces to engage and captivate users. Overall, the project served as a valuable learning experience, combining theoretical knowledge with practical implementation, and reinforcing the importance of adaptability and user-centric design in the realm of artificial intelligence.

**APPENDIX**

Code Implementation Details:The project involved the implementation of Rock, Paper, Scissors (RPS) game strategies using various AI approaches, including classical, pattern matching, rule-based, machine learning, and deep learning techniques. The code provided for each strategy is organized and documented to ensure clarity. The classical approach uses simple conditional statements and randomization. The pattern matching strategy utilizes a predefined set of moves to predict the user's next move based on historical data. The rule-based AI incorporates a more sophisticated decision-making process by analyzing the user's recent moves. The machine learning AI employs a decision tree classifier to predict the computer's move based on the user's previous choices. Finally, the deep learning AI utilizes a neural network model implemented using the Keras library.

Graphical User Interface (GUI) Development:The GUI development for the Rock, Paper, Scissors game is implemented using the Tkinter library in Python. The GUI features a window with labeled components such as player and AI sections, displaying the respective choices and game status. Images representing rock, paper, and scissors are loaded into the interface, providing a visual representation of the moves. User interaction is facilitated through buttons for each move, allowing players to make their selections. The layout is organized for clarity and ease of use, creating an intuitive and visually appealing environment for players. Additionally, the GUI incorporates a section to display the game's outcome, whether it's a win, loss, or tie, updating dynamically as the game progresses. The development emphasizes the integration of the GUI with the underlying game logic, ensuring a seamless and engaging user experience throughout the Rock, Paper, Scissors gameplay. The GUI development underscores the significance of user-centric design in AI applications, making them more accessible and enjoyable for users.

**REFERENCES**

* https://www.mdpi.com/2073-4336/12/3/52
* https://www.cs.stir.ac.uk/~kms/schools/rps/index.php
* https://www.essentially.net/rsp/play.jsp