Name: MD Ariful Islam

Email: arif\_9011@yahoo.com

**Rock Paper Scissors Game**

## Overview of the project

Rock-Paper-Scissors is a game in which each player simultaneously shows one of three hand signals representing rock, paper, or scissors. Rock beats scissors. Scissors beats paper. Paper beats rock. The player who shows the first option that beats the other player's option wins. This is an implementation of an interactive "Rock Paper Scissors" game in which the user can play with the computer using the camera.

## Learning objectives

The project's main learning objectives were the creation of a (small) image database for computer vision tasks, the set up of virtual environments and the installation of all required packages, and the practice of intermediate Python programming - especially 'if-else' statement, 'while' loops, and object oriented programming.

**MILESTONE 1: Set up environment**

* set up GitHub
* Create a Github repo

**MILESTONE 2: Create the computer vision system**

* creation of the dataset to be used to train the model used in the programme;
* creation of the model using [Teachable Machine](https://teachablemachine.withgoogle.com/).
* Create an image project model with four different classes: Rock,Paper,Scissors, Nothing and download the model.

**MILESTONE 3:  Installation of the dependencies.**

* creation of a new virtual environment;
* check the model working
* get familial with the code
* Install conda, PIP, tensorflow for python environment.
* Install ipykernel for using pip install ipykernel commmad.

**Code:**

import cv2

from keras.models import load\_model

import numpy as np

model = load\_model('keras\_model.h5')

cap = cv2.VideoCapture(0)

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

while True:

    ret, frame = cap.read()

    resized\_frame = cv2.resize(frame, (224, 224), interpolation = cv2.INTER\_AREA)

    image\_np = np.array(resized\_frame)

    normalized\_image = (image\_np.astype(np.float32) / 127.0) - 1 # Normalize the image

    data[0] = normalized\_image

    prediction = model.predict(data)

    cv2.imshow('frame', frame)

    # Press q to close the window

    print(prediction)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# After the loop release the cap object

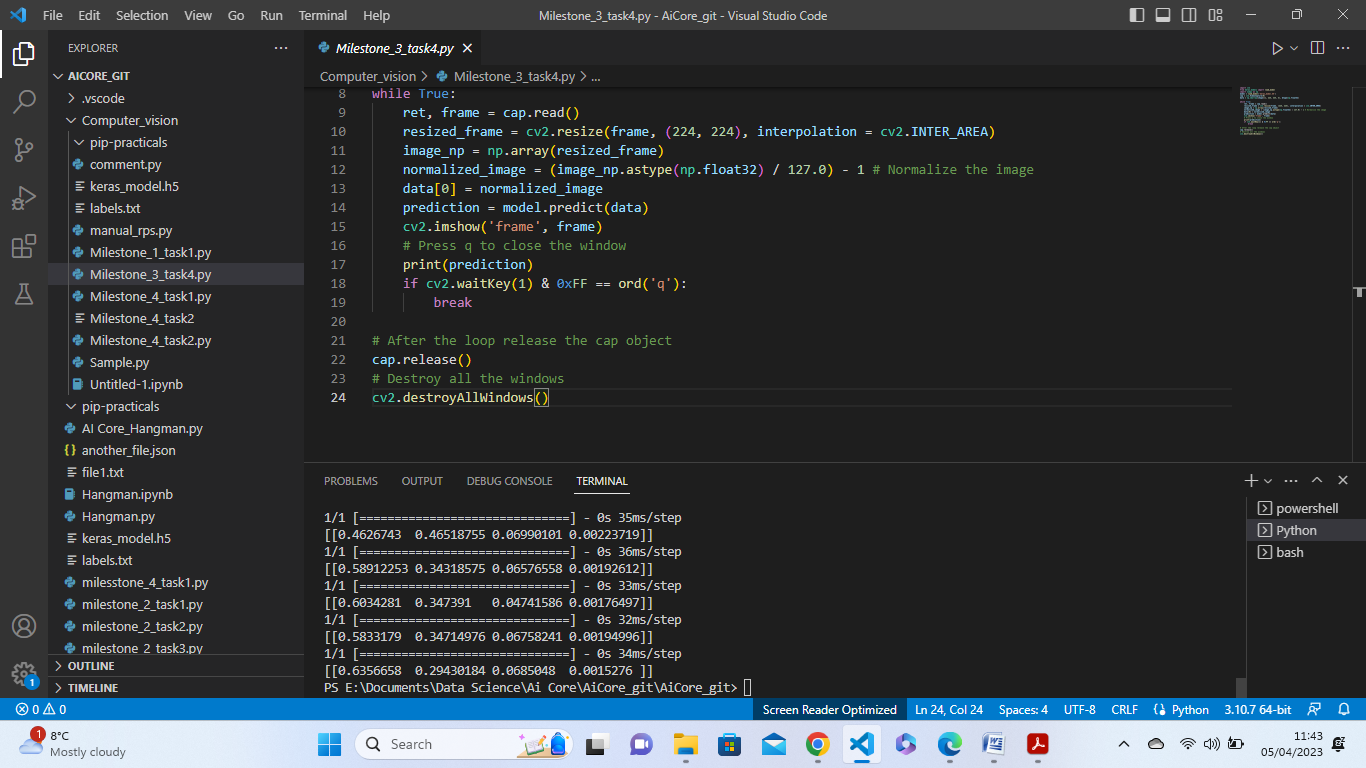
cap.release()

# Destroy all the windows

cv2.destroyAllWindows()

‘’’

Screenshot:



**4 - MILESTONE 4: Creation of a 'Rock, Paper, Scissors' game (manual\_rps.py).**

* store the user's and the computer's choices;
* figure out who won;
* create a function to simulate the game.

Create a file name manual\_rps.py for running game without camera. Then create two function get\_computer\_choice and get\_user\_choice. And use random module for choosing from rock, paper, scissors.

Use if else statement for selecting who will win the game based on the classic rule of rock paper and scissors. Create a function name get\_winner and computer\_choice and user\_choice pass as a parameters.

Create a function name play and I have called all three function inside this play function for running rock paper scissors game.

Code:

import random

class rock\_paper\_scissors:

    def \_\_init\_\_(self, computer\_list):

        self.computer\_list = computer\_list

        pass

    def get\_computer\_choice(self):

        computer\_choice = random.choice(self.computer\_list)

        return computer\_choice

    def get\_user\_choice(self):

        user\_choice = input("Enter your choice and play rock paper scissors game:")

        return user\_choice

    def get\_winner(self, computer\_choice, user\_choice):

        print(f"The computer picked {computer\_choice}, you picked {user\_choice}")

        if computer\_choice == "paper" and user\_choice == "rock" or computer\_choice == "scissors" and user\_choice == "paper" or computer\_choice == "rock" and user\_choice == "scissors":

            print ("You won!")

            pass

        elif computer\_choice == "rock" and user\_choice == "paper" or computer\_choice == "paper" and user\_choice == "scissors" or computer\_choice == "scissors" and user\_choice == "rock":

            print ("You lost")

            pass

        else:

            print("It is tie")

def play(computer\_list = ['rock','paper','scissors']):

    rps = rock\_paper\_scissors(computer\_list)

    computer\_choice = rps.get\_computer\_choice()

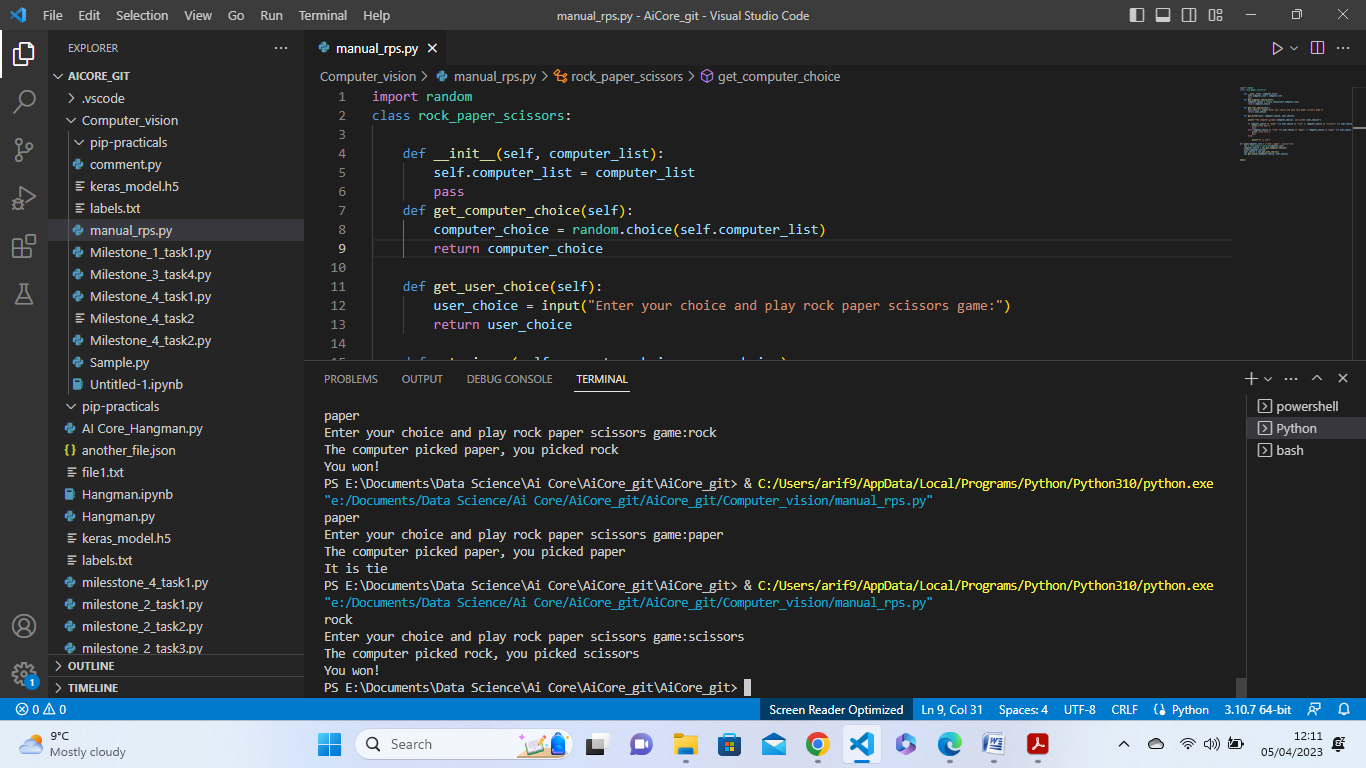
    print(computer\_choice)

    user\_choice = rps.get\_user\_choice()

    rps.get\_winner(computer\_choice, user\_choice)

play()

Screen Shoot:



**MILESTONE 5: Use the Camera to Play Rock-Paper-Scissors game (camera\_rps.py).**

* set up the camera and test the game;
* bonus implementations.

The camera version of the application gets the user choice using a webcam. The user is thus prompted to show a hand gesture to the camera, and the machine utilises keras\_model.h5 to guess the gesture and play the game accordingly.

## From user\_choice\_ to get\_prediction()

the manual version of the application, get\_user\_choice was a very simple method which utilised the input() function of the user\_choice attribute to obtain a textual prompt from the user, i.e., the chosen gesture:

The camera version is more complex and features three different methods that replace get\_user\_choice():

* get\_prediction(), which understands the user's input using keras\_model.h5 and probability;
* classify\_output, which uses the list of probabilities from get\_prediction() to determine the image inputted in the camera.

Code:

    def countdown():

        countdown = 5

        print("\nGet ready to show your choice :")

        while countdown >= 0:

            print(f'{countdown}')

            cv2.waitKey(1000)

            countdown -= 1

        print("\nNow show your hand choice :")

    countdown()

    def get\_prediction(self):

        model = load\_model("keras\_Model.h5", compile=False)

        cap = cv2.VideoCapture(0)

        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

        end\_time = time.time() + 5

        while time.time() < end\_time:

            ret, frame = cap.read()

            if ret==False:

                continue

            resized\_frame = cv2.resize(frame, (224, 224), interpolation = cv2.INTER\_AREA)

            image\_np = np.array(resized\_frame)

            normalized\_image = (image\_np.astype(np.float32) / 127.0) - 1 # Normalize the image

            data[0] = normalized\_image

            prediction = model.predict(data)

            cv2.imshow('frame', frame)

            print(prediction)

            index = np.argmax(prediction)

            print(index)

            if index == 0:

                choice=="rock"

                print("rock")

            elif index == 1:

                choice="paper"

                print("paper")

            elif index == 2:

                choice="scissors"

                print("scissors")

            elif index == 3:

                choice="nothing"

                print("nothing")

            if (cv2.waitKey(1) & 0xFF == ord('q')):

                break

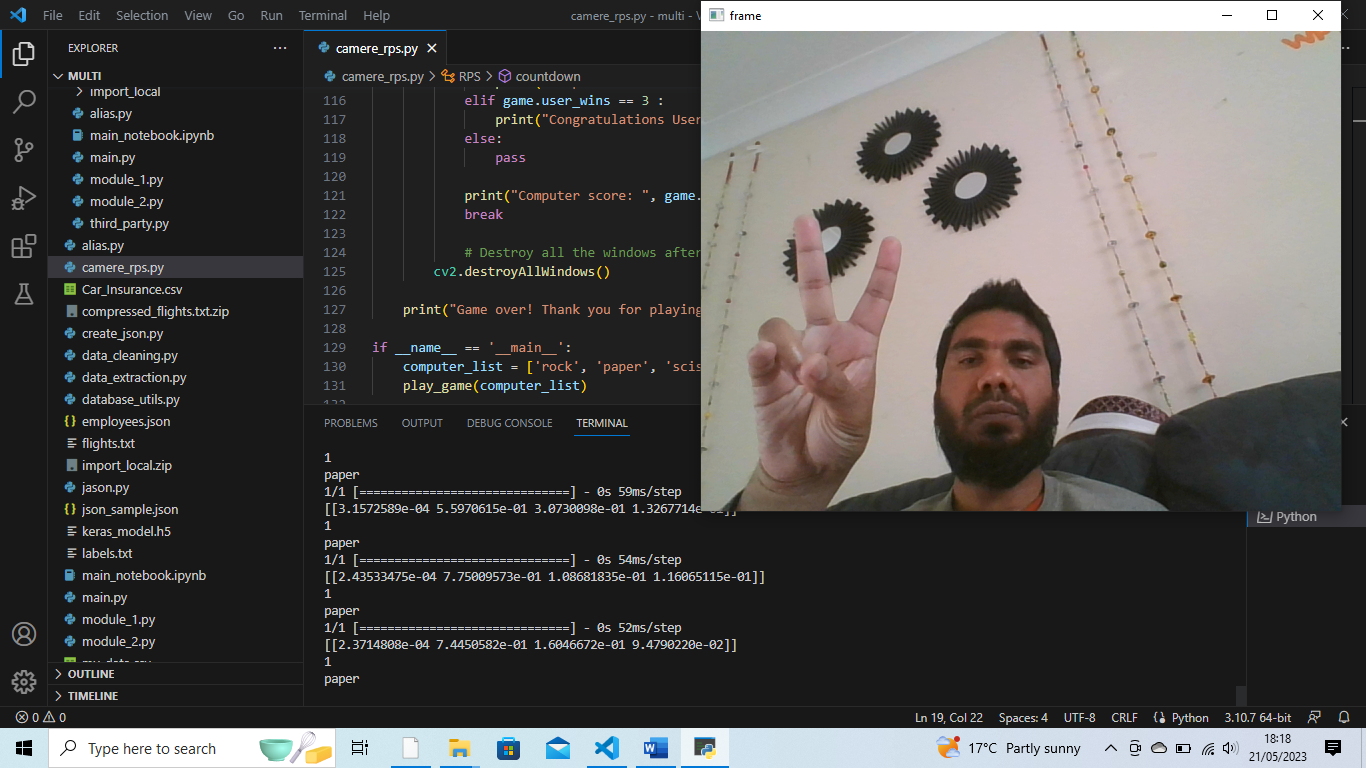
        cap.release()

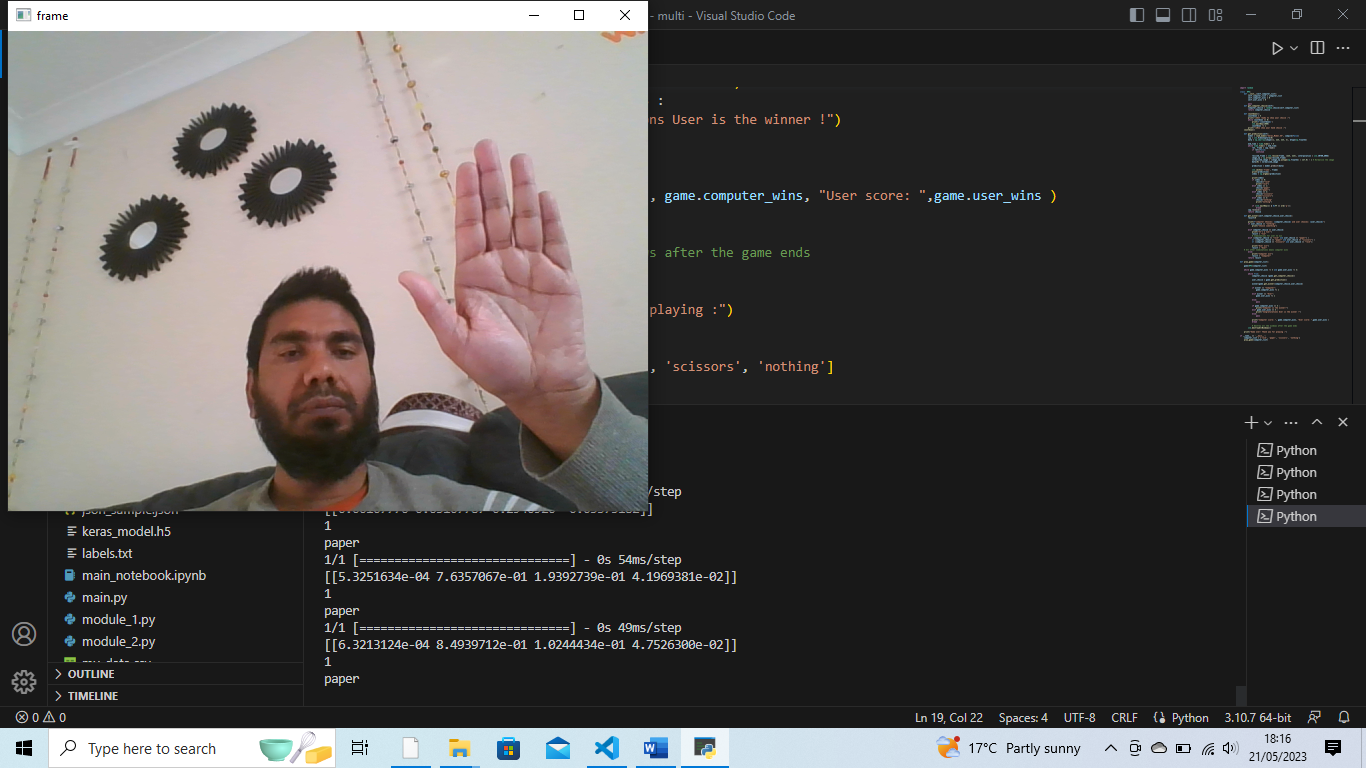
        return choice

**Count Down**

countdown(), this function use time.time() for making countdown. When Countdown function start countdown from 5 seconds to 0 then start camera for showing hand to gesture. The former slows down the machine to make the programme user-friendly and asks the user to prepare to show their hand, while the latter merely slows down the programme and prints two rows of full stops when needed. The effects of these can be seen in the image below.

Screen Shoot:





**Function get\_winner()**

Function get\_winner has been user for who wins the game computer or user. Computer will choice randomly from giver list of Rock Paper Scissors user will show hand to gesture for choosing input.

Code:

def get\_winner(self,computer\_choice,user\_choice):

        result=0

        print(f"computer Chooses: {computer\_choice} and user chooses: {user\_choice}")

        if user\_choice == "nothing":

            print("Choose something")

        elif computer\_choice == user\_choice:

            print("It's a tie!")

            result = "tie"

            # Combinations for user to win

        elif (computer\_choice == "rock" and user\_choice == "paper") \

            or (computer\_choice == "paper" and user\_choice == "scissors") \

            or (computer\_choice == "scissors" and user\_choice == "rock"):

            print("User win")

            result = "User"

    # All other Combinations means computer wins

        else:

            print("Computer win")

            result = "Computer"

        return result

**Function play\_game()**

Fuction play\_game() is the main function of the whole game. It’s repeat three time for finding winner score 3. So, if user gets overall score 3 then user will win. It’s print(“Congratulations user is the winner!”). Same if the computer gets overall score 3 then computer will win. It’s print(“Computer is the winner!”).

Code:

def play\_game(computer\_list):

    game=RPS(computer\_list)

    while game.computer\_wins != 3 and game.user\_wins != 3:

        while True:

            computer\_choice =game.get\_computer\_choice()

            user\_choice = game.get\_prediction()

            winner=game.get\_winner(computer\_choice,user\_choice)

            if winner == "Computer":

                game.computer\_wins += 1

            elif winner == "User":

                game.user\_wins += 1

            else:

                pass

            if game.computer\_wins == 3 :

                print("Computer is the winner!")

            elif game.user\_wins == 3 :

                print("Congratulations User is the winner !")

            else:

                pass

            print("Computer score: ", game.computer\_wins, "User score: ",game.user\_wins )

            break

            # Destroy all the windows after the game ends

        cv2.destroyAllWindows()

    print("Game over! Thank you for playing :")

**function if \_\_name\_\_== ‘\_\_main\_\_’:**

if \_\_name\_\_== ‘\_\_main\_\_’, function is used for calling play\_game() for playing whole game.

Code:

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

    computer\_list = ['rock', 'paper', 'scissors', 'nothing']

    play\_game(computer\_list)

# **Finally, What I have learned**

* Create a teachable machine model that recognises different hand signals for the game.
* Create a virtual working environment using conda and install required libraries.
* Create a manual RPS game where the user and the computer choose their option and the code decided the winner.
* Create a camera RPS game that captures the choice of the user and decided the winner by comparing it with the choice of the computer.