AI Project Part 2 Report – i200983 – CS(B) – Fahad Kamran

Firstly, I tried to gather data in order to train the ai model for the fight function of the 'bot.py' rather than the use of if statements. Using a model rather than if statements improves traversal time of the function and increase conditional qualities when the game is executed.

In order to do this I started with a writing to csv codes in both 'controller.py' and 'bot.py'

Code in controller.py (inside def main at the start):

```
csv_file = 'GameData.csv'

    # Write data to the CSV file
    # Append data to the CSV file
    with open(csv_file, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([ 'Timer', 'fight_result',
                          'has_round_started', 'is_round_over','player1_id',
'player1_health', 'player1_x_coord', 'player1_y_coord',
                          'player1_is_jumping', 'player1_is_crouching',
'player1_is_player_in_move',
                          'player1_move_id', 'player2_id', 'player2_health',
'player2_x_coord',
                          'player2_y_coord', 'player2_is_jumping',
'player2_is_crouching',
                          'player2_is_player_in_move', 'player2_move_id',
'player1_button_up',
                          'player1_button_down', 'player1_button_right',
                          'player1_button_left', 'player1_button_select',
                          'player1_button_start', 'player1_button_Y',
                          'player1_button_B', 'player1_button_X',
                          'player1_button_A', 'player1_button_L',
                          'player1_button_R', 'player2_button_up',
                          'player2_button_down', 'player2_button_right',
                          'player2_button_left', 'player2_button_select',
                          'player2_button_start', 'player2_buttons.Y',
                          'player2_button_B', 'player2_button_X',
                          'player2_button_A', 'player2_button_L',
                          'player2_button_R'])
```

Code in the bot.py (inside the def fight function at the start):

```
 # Specify the CSV file path
        csv_file = 'GameData.csv'

        # Write data to the CSV file
       # Append data to the CSV file
        with open(csv_file, 'a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([
```

```
            current_game_state.timer,
            current_game_state.fight_result,
            current_game_state.has_round_started,
            current_game_state.is_round_over,
            current_game_state.player1.player_id,
            current_game_state.player1.health,
            current_game_state.player1.x_coord,
            current_game_state.player1.y_coord,
            current_game_state.player1.is_jumping,
            current_game_state.player1.is_crouching,
            current_game_state.player1.is_player_in_move,
            current_game_state.player1.move_id,
            current_game_state.player2.player_id,
            current_game_state.player2.health,
            current_game_state.player2.x_coord,
            current_game_state.player2.y_coord,
            current_game_state.player2.is_jumping,
            current_game_state.player2.is_crouching,
            current_game_state.player2.is_player_in_move,
            current_game_state.player2.move_id,
            current_game_state.player1.player_buttons.up,
            current_game_state.player1.player_buttons.down,
            current_game_state.player1.player_buttons.right,
            current_game_state.player1.player_buttons.left,
            current_game_state.player1.player_buttons.select,
            current_game_state.player1.player_buttons.start,
            current_game_state.player1.player_buttons.Y,
            current_game_state.player1.player_buttons.B,
            current_game_state.player1.player_buttons.X,
            current_game_state.player1.player_buttons.A,
            current_game_state.player1.player_buttons.L,
            current_game_state.player1.player_buttons.R,
            current_game_state.player2.player_buttons.up,
            current_game_state.player2.player_buttons.down,
            current_game_state.player2.player_buttons.right,
            current_game_state.player2.player_buttons.left,
            current_game_state.player2.player_buttons.select,
            current_game_state.player2.player_buttons.start,
            current_game_state.player2.player_buttons.Y,
            current_game_state.player2.player_buttons.B,
            current_game_state.player2.player_buttons.X,
            current_game_state.player2.player_buttons.A,
            current_game_state.player2.player_buttons.L,
            current_game_state.player2.player_buttons.R
                    ])

        print("Data successfully stored in data.csv.")
```

These lines of code directly above the rest of the code in the fight function.

Afterwards, we run the game once as stated in the ‘readme.txt’ and play one round. Come back to the controller.py and comment out the code we added and continue to run the game until we have enough data to make a model with. We then comment out the code we added to fight function in the bot.py and run ‘project.ipynb’ which will create the model and scalar pickle file to be used on later.
The ‘project.ipynb’ file loads the data into variable using pandas and grounds them according to their column name as written in the first row of the ‘GameData.csv’. Null values are removed and data is mapped to fit numerical values as they need to be for model creation using dictionary. Using sklearn library, the neural network model is created.

Next new code is added to bot.py to add the model pickle files for use, new code for buttons prediction and comment out the previous code of the fight function.

```
def fight(self,current_game_state,player):
        #python Videos\gamebot-competition-master\PythonAPI\controller.py 1
        #start of my code

        if player == "1":
            if self.exe_code != 0:
                self.run_command([], current_game_state.player1)

            # Prepare the input data for prediction
            input_data = np.array([
            current_game_state.player1.player_id,
            current_game_state.player1.health,
            current_game_state.player1.x_coord,
            current_game_state.player1.y_coord,
            current_game_state.player1.is_jumping,
            current_game_state.player1.is_crouching,
            current_game_state.player1.player_buttons,
            current_game_state.player1.is_player_in_move,
            current_game_state.player1.move_id
        ]).reshape(1, -1)
            # Apply the same scaling as used during training
            input_data = scaler.transform(input_data)
            # Predict the button inputs using the model
            predicted_buttons = model.predict(input_data)

            # Set the predicted buttons as the player's buttons
            self.my_command.player_buttons = predicted_buttons

        elif player == "2":
            if self.exe_code != 0:
                self.run_command([], current_game_state.player2)

            # Prepare the input data for prediction
            input_data = np.array([
```

```
            current_game_state.player2.player_id,
            current_game_state.player2.health,
            current_game_state.player2.x_coord,
            current_game_state.player2.y_coord,
            current_game_state.player2.is_jumping,
            current_game_state.player2.is_crouching,
            current_game_state.player2.player_buttons,
            current_game_state.player2.is_player_in_move,
            current_game_state.player2.move_id
        ]).reshape(1, -1)
            # Apply the same scaling as used during training
            input_data = scaler.transform(input_data)
            # Predict the button inputs using the model
            predicted_buttons = model.predict(input_data)

            # Set the predicted buttons as the player's buttons
            self.my_command.player2_buttons = predicted_buttons

        return self.my_command
```

This is the code of the fight function that uses the input data, transforms it using the scalar model and makes the predictions of the buttons using neural network model.

After this we run the game again, this time the game will use the neural network model to predict moves rather than if statements for more likely to win than before.