AI Project - Street Fighter 2 Bot

I201866 Faraz Razi - I201893 Ahmed

Section F

Report

Data Preparation:

We collected a dataset containing the game states and corresponding moves. Ensure that the dataset includes both player 1's moves and the resulting moves made by player 2.
At time of submission we trained our models with a sufficient amount of data. We will be adding and generating more data for future tournaments.

Feature Engineering:

Extract meaningful features from the available columns that can effectively capture the game state. We opted for following features from the data

Timer: The remaining time in the game.
Opponent character: The character chosen by the opponent.
Opponent health: The remaining health of the opponent.
Opponent movement: Whether the opponent is jumping, crouching, or in motion.
Opponent button presses: The buttons pressed by the opponent.
Player character: Your character in the game.
Player health: Your remaining health.
Player movement: Your movement state (jumping, crouching, or in motion).
Player button presses: The buttons you press.
Position differences: The differences in X and Y coordinates between you and the opponent.You can further enhance these features or add additional ones based on the specifics of your game.

Data Preprocessing:

Preprocess the dataset by normalizing numeric features, encoding categorical features (such as character names), and splitting the data into training and testing sets.

The Data Preprocessing step includes several data preprocessing steps to ensure the data is in the appropriate format for analysis. The following preprocessing steps are performed:

- Removal of Rows: Rows where the 'round_started' column is False are removed from the dataset. These rows represent rounds that have not yet started and are irrelevant for analysis.
- Removal of Extra Columns: Columns such as 'result', 'round_started', 'round_over', 'p1_Select', 'p1_Start', 'p2_Select', and 'p2_Start' are dropped from the dataset as they are not necessary for the analysis.
- Player Identification: The function identifies the player in control by examining the 'player' column in the first row. The player's moves and character information are then appropriately labeled as 'player' or 'opponent' based on this identification.
- Flipping X Coordinates: If the player is identified as player 1, the function flips the X coordinates for both the player and the opponent. This adjustment is made to ensure consistency in the coordinate system across different player positions.
- Calculation of Coordinate Differences: The function calculates the difference between the X and Y coordinates of the player and the opponent. These differences are stored in the 'x_diff' and 'y_diff' columns, respectively.
- Conversion of Moves to Numbers: The function converts the move inputs (e.g., Up, Down, Right, Left, etc.) for both the player and the opponent into binary integers. Each move is assigned a binary value (0 or 1) to represent its presence or absence.
- Combination of Moves: The function combines the binary move inputs for the player and the opponent into unique move IDs. These move IDs are created by concatenating the binary values of each move into a single string. The resulting move IDs are stored in the 'player_moves' and 'opponent_moves' columns.
- Conversion to Binary Integers: The move IDs in the 'player_moves' and 'opponent_moves' columns are converted from binary strings to binary integers using the int(x, 2) function. This conversion enables numerical representation of the move IDs for further analysis.
- Removal of Redundant Columns: The columns related to individual move inputs are dropped from the dataset as they are no longer needed after the combination and conversion steps.
- Conversion of Remaining Columns: The remaining columns, such as 'player_jumping', 'player_crouching', 'player_in_move', 'opponent_jumping', 'opponent_crouching', and 'opponent_in_move', are converted to binary integers to ensure consistency in data types.
- Removal of Player Identification Columns: The 'player' column, 'p1_move' column, and 'p2_move' column are dropped from the dataset, as they are no longer necessary after the preprocessing steps.

Model Selection:

In our case we decided to use the decision tree classifier as the model for analysis. Model selection involved an appropriate algorithm or model architecture based on the problem at hand and the available data. Other models could also be considered depending on the nature of the data and the specific objectives of the analysis.

Prediction and Integration:

Once the model is trained and evaluated, we use it to predict player 2's counter moves based on the current game state provided as input. Integrated the model predictions into the game logic to generate suitable counter moves for player 2.

The integration of the model into the game allows for dynamic decision-making based on the current game state. The playModel function processes each frame, predicts an action using the decision tree classifier model, and executes the action accordingly. The model's predictions enable the player to make informed decisions based on the game environment. The learning aspect is also incorporated, as the current game state is saved for potential learning and improvement. Overall, this integration facilitates the integration of AI-based decision-making into the game environment, enhancing player experience and gameplay.