**Artificial Intelligence Game Bot Project Report**

**Section: DS-A**

**Group Members**

* **21I-2699 Muhammad Sharjeel Nadir**
* **21I-1777 Areeba**
* **21I-1707 Romaisa**
* **21I-1702 Masroor Bin Rehan**

**Project Objective**

**The aim of this project is to design an AI-powered bot that plays *Street Fighter II Turbo* intelligently by analyzing real-time game states and predicting optimal actions using a trained machine learning model. The default rule-based logic provided in the starter code was replaced with a data-driven classification model that learns from actual gameplay.**

**Dataset and Features**

**Data Source**

**The dataset (game\_data.csv) was generated by manually playing matches using every character in the game. During these matches, we replaced the ML bot logic in bot.py with a custom data-collecting version (bot\_for\_data\_collection.py). Each frame of gameplay was logged, resulting in thousands of samples.**

**Feature Description**

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| **health** | **int** | **Player's remaining health** |
| **x, y** | **int** | **Player's coordinates on the screen** |
| **is\_jumping** | **int** | **Whether the player is jumping (0/1)** |
| **is\_crouching** | **int** | **Whether the player is crouching (0/1)** |
| **move\_id** | **int** | **ID of the move the player is executing** |
| **enemy\_health** | **int** | **Opponent's health** |
| **enemy\_x, enemy\_y** | **int** | **Opponent's coordinates** |
| **action** | **object** | **Target label: move performed by player** |

**Class Imbalance**

**The action column is highly imbalanced. Common actions like far\_combo\_1 had over 1000 samples, while rare actions like close\_combo\_2 had less than 20. This imbalance can cause classifiers to bias toward majority classes.**

**Preprocessing & Balancing**

**Label Encoding**

**To train machine learning models, the action column was label-encoded using LabelEncoder to convert class names into numeric values.**

**Class Balancing Techniques Explored**

**To resolve the class imbalance, we tested three sampling techniques:**

1. **SMOTE (Synthetic Minority Oversampling Technique)  
   → Generates new synthetic samples for minority classes.**
2. **ADASYN (Adaptive Synthetic Sampling)  
   → Similar to SMOTE but focuses on harder-to-learn samples.**
3. **SMOTEENN (SMOTE + Edited Nearest Neighbors)   
   → Combines SMOTE with a cleaning step that removes noisy or ambiguous samples using nearest-neighbor analysis.**

**Why SMOTEENN?**

**Out of all techniques tested, SMOTEENN delivered the highest accuracy of 97.9%. It worked well because:**

* **It oversampled rare actions effectively (like walk\_right, close\_combo\_1).**
* **It cleaned overlapping samples using ENN, improving decision boundaries.**

**Models Evaluated**

**We trained and evaluated the following classification models using balanced data:**

| **Model** | **Accuracy** |
| --- | --- |
| **Random Forest** | **61.9% (before SMOTEENN)** |
| **XGBoost (GPU-accelerated)** | **61.4%** |
| **MLP (Neural Network)** | **44.1%** |
| **Logistic Regression** | **28.6%** |

**After applying SMOTEENN, Random Forest reached 97.9% accuracy, making it the model of choice for deployment.**

**Folder Structure**

**├── bot.py # ML-powered action prediction**

**├── bot\_for\_data\_collection.py # Logs gameplay into CSV**

**├── controller.py # Game loop (single & two-player mode)**

**├── model\_training.ipynb # Training + evaluation notebook**

**├── model.pkl # Final trained model (Random Forest)**

**├── label\_encoder.pkl # Encoded labels for prediction**

**├── game\_data.csv # Collected training data**

**├── buttons.py, command.py, game\_state.py, player.py**

**Full Workflow Summary**

**Step 1: Start the Game**

* **Open EmuHawk.exe from the provided BizHawk emulator.**
* **File → Open ROM → Select Street Fighter II Turbo (U).smc.**
* **Open Toolbox (Shift+T). Leave it running.**

**Step 2: Collect Data**

* **Copy code from bot\_for\_data\_collection.py into bot.py.**
* **Play as every character against the CPU.**
* **This generates a training dataset game\_data.csv.**

**Step 3: Train the Model**

* **Open model\_training.ipynb in Jupyter.**
* **Run all cells to:**
  + **Load and balance data using SMOTEENN.**
  + **Train Random Forest model.**
  + **Save files:**
    - **model.pkl: The trained classifier.**
    - **label\_encoder.pkl: Used to map class labels during prediction.**

**Step 4: Run the Bot**

* **Restore original bot.py (ML logic).**

**python controller.py 1**

* **The bot will now play intelligently as Player 1.**

**Two-Player Mode (Bot vs Bot)**

1. **Launch two terminal windows.**
2. **In one terminal:**

**python controller.py 1**

1. **In the other:**

**python controller.py 2**

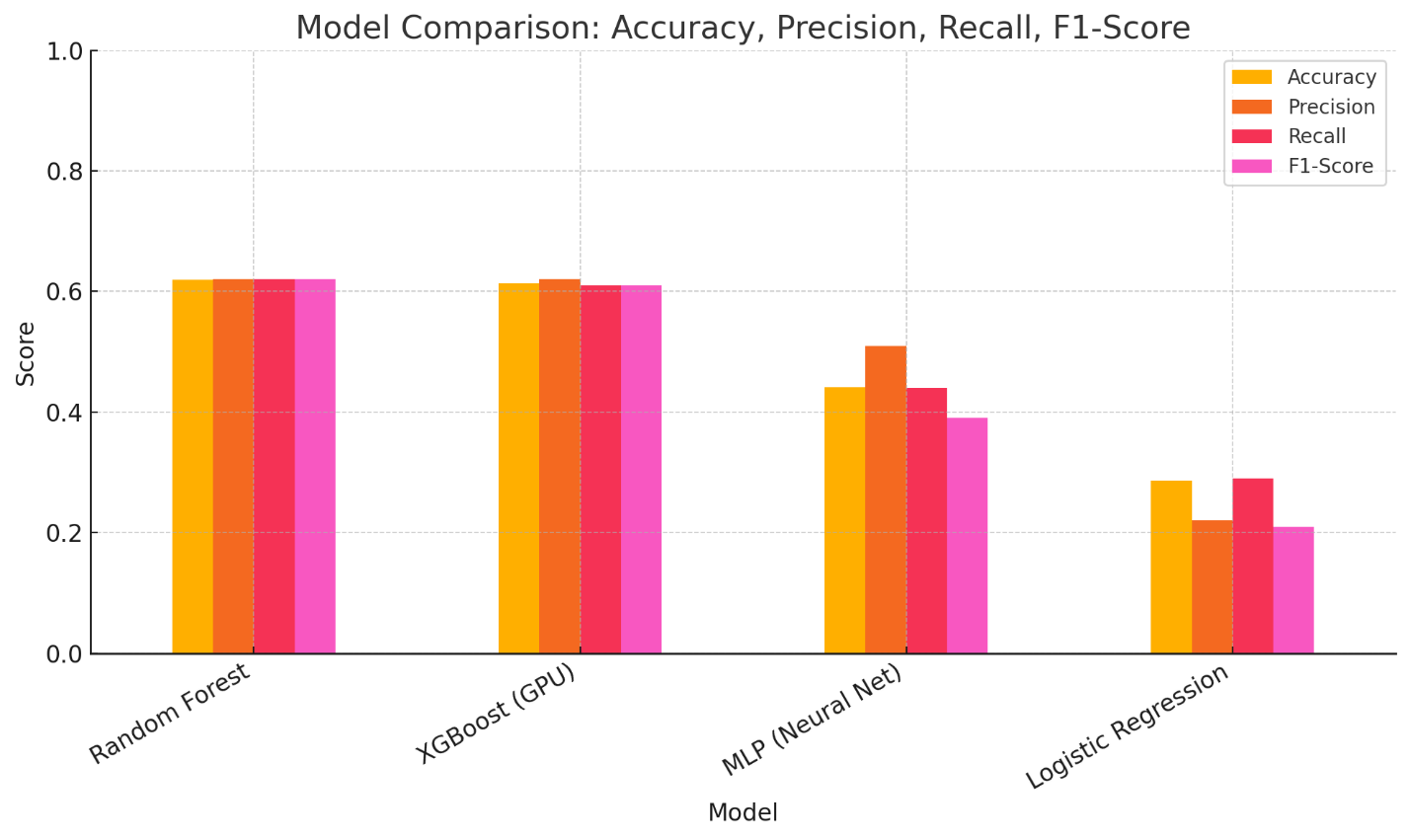
1. **In the game, go to VS Battle Mode.**
2. **Click the Gyroscope Bot icon to start the fight.**

**Final Result**

**After trying multiple classifiers and balancing techniques, we chose Random Forest + SMOTEENN, achieving:**

* **97.9% accuracy**
* **Near-perfect F1-scores across all actions**
* **Reliable predictions even for rare actions**

**Comparison chart:**

****