Multi-Arm-Bandit-Agent

Homework at Multi-Agent Learning course

Introduction

This application simulates the Multi-Arm Bandit problem using different algorithms, including the UCB1 strategy and the Epsilon-Greedy strategy. The Multi-Arm Bandit problem is a classic problem in decision-making and reinforcement learning, where an agent must choose between multiple actions (arms) to maximize its cumulative reward over time.

Installation

To run the simulation, follow these steps:

- Clone the repository to your local machine.

```
git clone https://github.com/your-username/Multi-Arm-Bandit-Agent.git
```

```
cd Multi-Arm-Bandit-Agent
```

- Ensure you have ***Python 3.8*** installed on your system.

  Make sure you have Python installed on your system. You can download it from python.org.

- Create a Virtual Environment (Optional but Recommended).

```
python -m venv venv
```

```
source venv/bin/activate # On Windows, use: .\venv\Scripts\activate
```

Activating the virtual environment isolates the project dependencies.

- Navigate to the project directory in the terminal.

```
cd /path/to/Multi-Arm-Bandit-Agent
```

- Install the required dependencies:

  Install the necessary libraries listed in ***requirements.txt***:

```
pip install -r requirements.txt
```

- Run the GUI application:

  Execute the following command to launch the GUI application:

```
python main.py
```

Application Overview

The application includes a graphical user interface (GUI) that allows users to:

Load a data file containing the number of arms, total iterations, and epsilon values. Run the bandit simulation based on the provided data. Visualize the results with a plot showing the average rewards over iterations for UCB1 and Epsilon-Greedy strategies.

Technical Details

Algorithms

- **UCB1 Strategy:**

  The UCB1 strategy selects arms based on the Upper Confidence Bound algorithm, balancing exploration and exploitation. The formula for selecting an arm is based on the estimated average reward and an exploration bonus.

- **Epsilon-Greedy Strategy:**

  The Epsilon-Greedy strategy selects arms with a probability of epsilon for exploration and with a probability of (1 - epsilon) for exploitation. It exploits the arm with the highest average reward.

Implementation

- The application is implemented in Python using the Tkinter library for the GUI and matplotlib for plotting.
- The core algorithms for UCB1 and Epsilon-Greedy agents are implemented in separate classes (UCB1Agent and EpsilonGreedyAgent).
- The Multi-Armed Bandit problem is modeled using the MultiArmedBandit class.

Using the GUI:

- **Load Data File:**

  Click the "Browse" button to select a data file. The data file should contain the number of arms, total iterations, and epsilon values.

- **Run Simulation:**

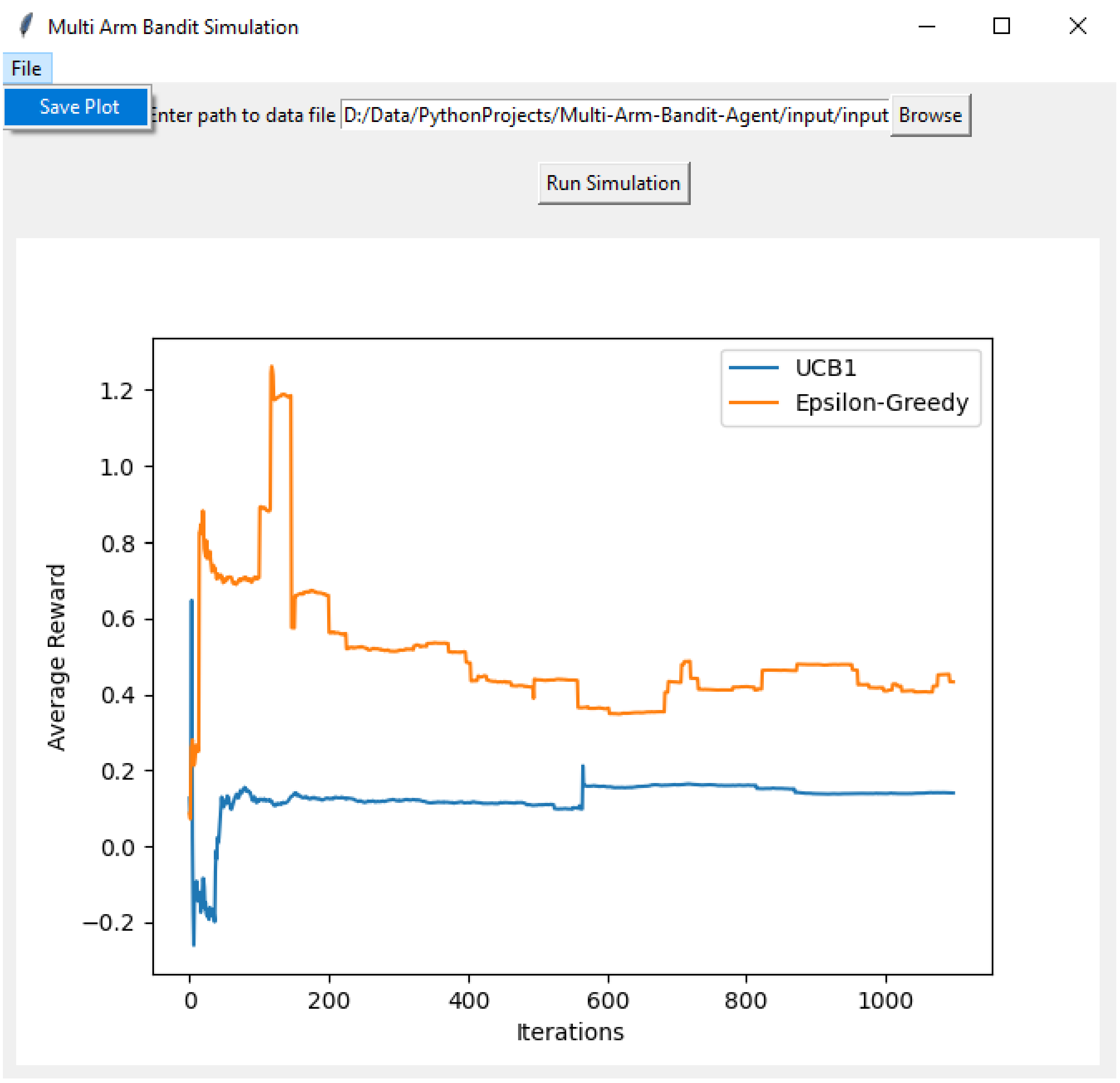
  After loading the data file, click the "Run Simulation" button to start the bandit simulation.

- **View Results:**

  The GUI will display a plot showing the average rewards over iterations for both UCB1 and Epsilon-Greedy strategies.

- **Save Plot (Optional):**

  In the menu bar, go to "File" and select "Save Plot" to save the generated plot to the output directory.

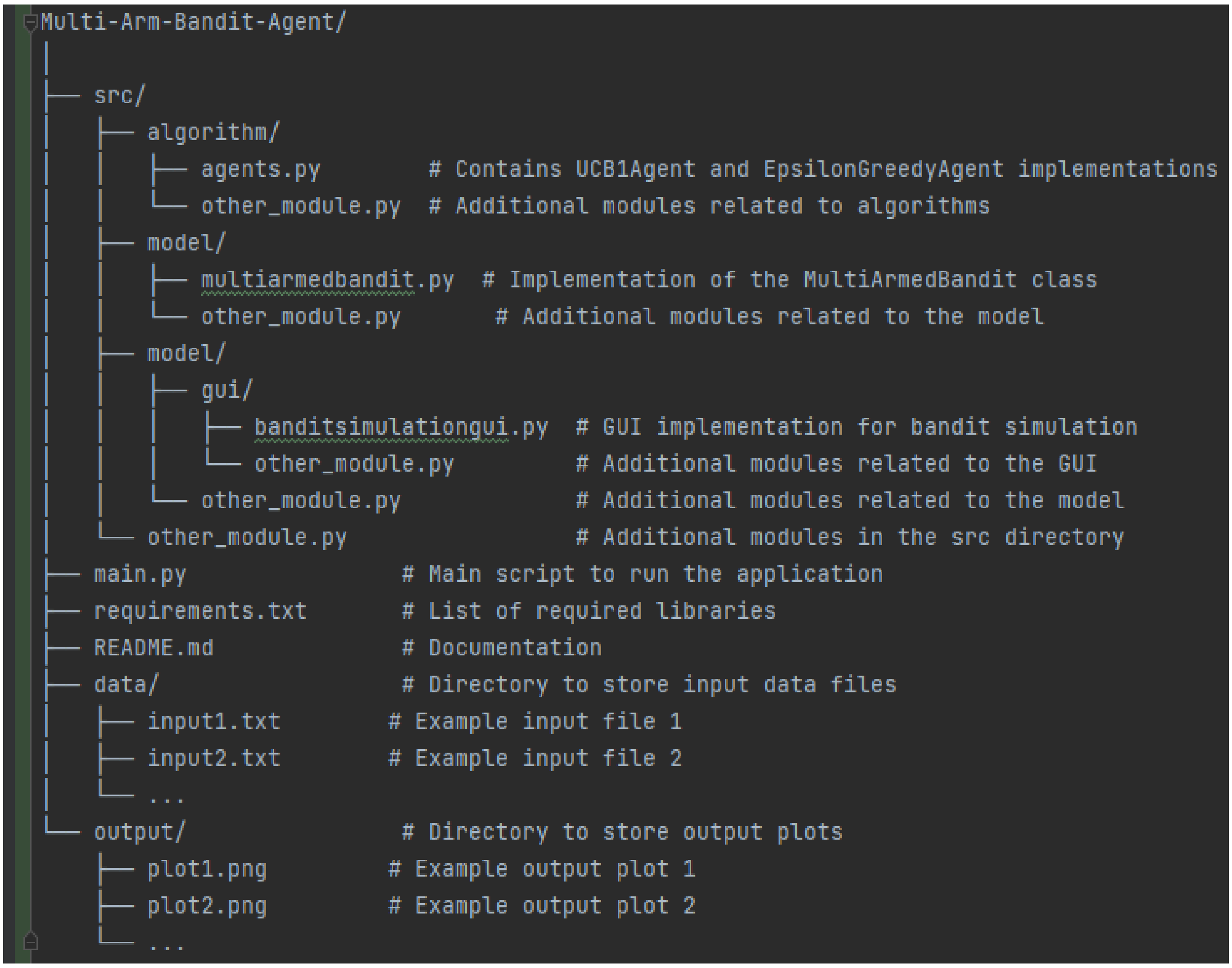


Output Structure

The output of the simulation is a plot displaying the average rewards over iterations for both the UCB1 and Epsilon-Greedy strategies. Each run of the simulation generates a unique output plot, and the plots are saved in the output directory.

Folder Structure:

The project folder structure is as follows:



```
Multi-Arm-Bandit-Agent/
│
├── src/
│   ├── algorithm/
│   │   ├── agents.py        # Contains UCB1Agent and EpsilonGreedyAgent implementations
│   │   └── other_module.py  # Additional modules related to algorithms
│   ├── model/
│   │   ├── multiarmedbandit.py  # Implementation of the MultiArmedBandit class
│   │   └── other_module.py       # Additional modules related to the model
│   ├── model/
│   │   ├── gui/
│   │   │   ├── banditsimulationgui.py  # GUI implementation for bandit simulation
│   │   │   └── other_module.py         # Additional modules related to the GUI
│   │   └── other_module.py             # Additional modules related to the model
│   └── other_module.py                 # Additional modules in the src directory
├── main.py                 # Main script to run the application
├── requirements.txt        # List of required libraries
├── README.md               # Documentation
├── data/                   # Directory to store input data files
│   ├── input1.txt        # Example input file 1
│   ├── input2.txt        # Example input file 2
│   └── ...
└── output/                 # Directory to store output plots
    ├── plot1.png         # Example output plot 1
    ├── plot2.png         # Example output plot 2
    └── ...
```

Notes:

- ***Virtual Environment (Optional):***

  Activating the virtual environment (venv) is optional but recommended to maintain a clean and isolated Python environment for the project.

- ***Data Files:***

  Ensure that your data files are formatted correctly, with the number of arms, total iterations, and epsilon values specified.

- ***Output:***

  Output plots will be saved in the output directory with unique timestamps.