Multi-Arm-Bandit-Agent

Homework at Multi-Agent Learning course ---> To implement an agent that wants to optimize the operation of a multi-slot machine (multi-arm bandit) with several actuation arms. The agent will experience at least 2 learning strategies taken from specialized literature.

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

Step 1: Install Python

Download Python: Visit python.org. and download the 3.8 version of Python for your operating system.

Install Python: Follow the installation instructions provided on the Python website. During installation, make sure to check the option to add Python to your system PATH.

Step 2: Install PyCharm

Download PyCharm: Visit JetBrains PyCharm download and download the Community Edition, which is free.

Install PyCharm: Follow the installation instructions provided on the PyCharm website.

Step 3: Clone the Repository

Open Terminal/Command Prompt: Open your terminal or command prompt.

Clone the Repository to your local machine: Run the following commands to clone the repository:

```
git clone https://github.com/AndreeaDraghici/Multi-Arm-Bandit-Simulation-Agent.git
```

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

Step 4: Create and Activate Virtual Environment (Optional)

Create a Virtual Environment (Optional but Recommended).

Create Virtual Environment using conda environment: If you want to create a virtual environment, run the following commands:

```
conda create --name myenv python=3.7
```

```
conda activate myenv
```

Activating the virtual environment isolates the project dependencies.

Step 5: Install Dependencies

Navigate to the Project Directory: Open a terminal/command prompt and go to the project directory:

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

Install Dependencies:

Make sure to install the required dependencies.

To install the necessary libraries you have two possibility:

1. Run the following command in the terminal to install the required dependencies:

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

2. To install manual the libraries in Python, you can use the pip install <library_name> command like below.

E.g: `pip install numpy`

Step 6: Configure Logging

Ensure that the logging configuration file (logging_config.yml) is set up according to your preferences. This file is used to configure logging in the tool.

Step 7: Run the GUI Application

Execute the following command to launch the GUI application:

```
python main.py
```

Alternative Step

Or, application provides an executable (***exe***) file in the '*dist*' directory, users can follow these alternative steps to run the application:

Navigate to the "*dist*" directory: Open a terminal or File Explorer and go to the "*dist*" folder within your project directory.

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

Run the Executable: Execute the following command to run the executable file:

```
./multi_arm_agent.exe # On Windows, simply double-click the executable file
```

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:**

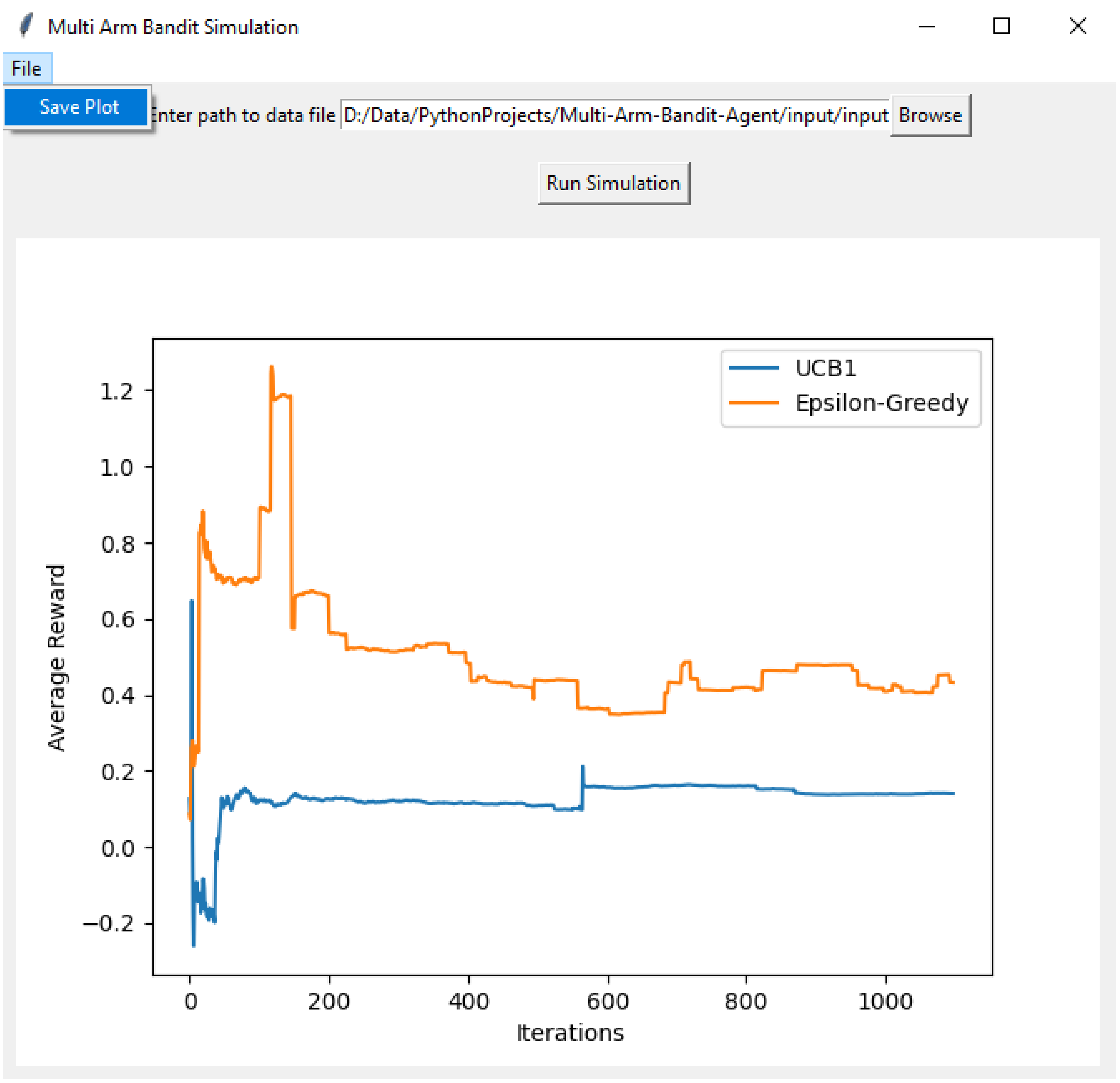
  After loading the data file, click the "Run Simulation" button to start the bandit simulation.

- **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.

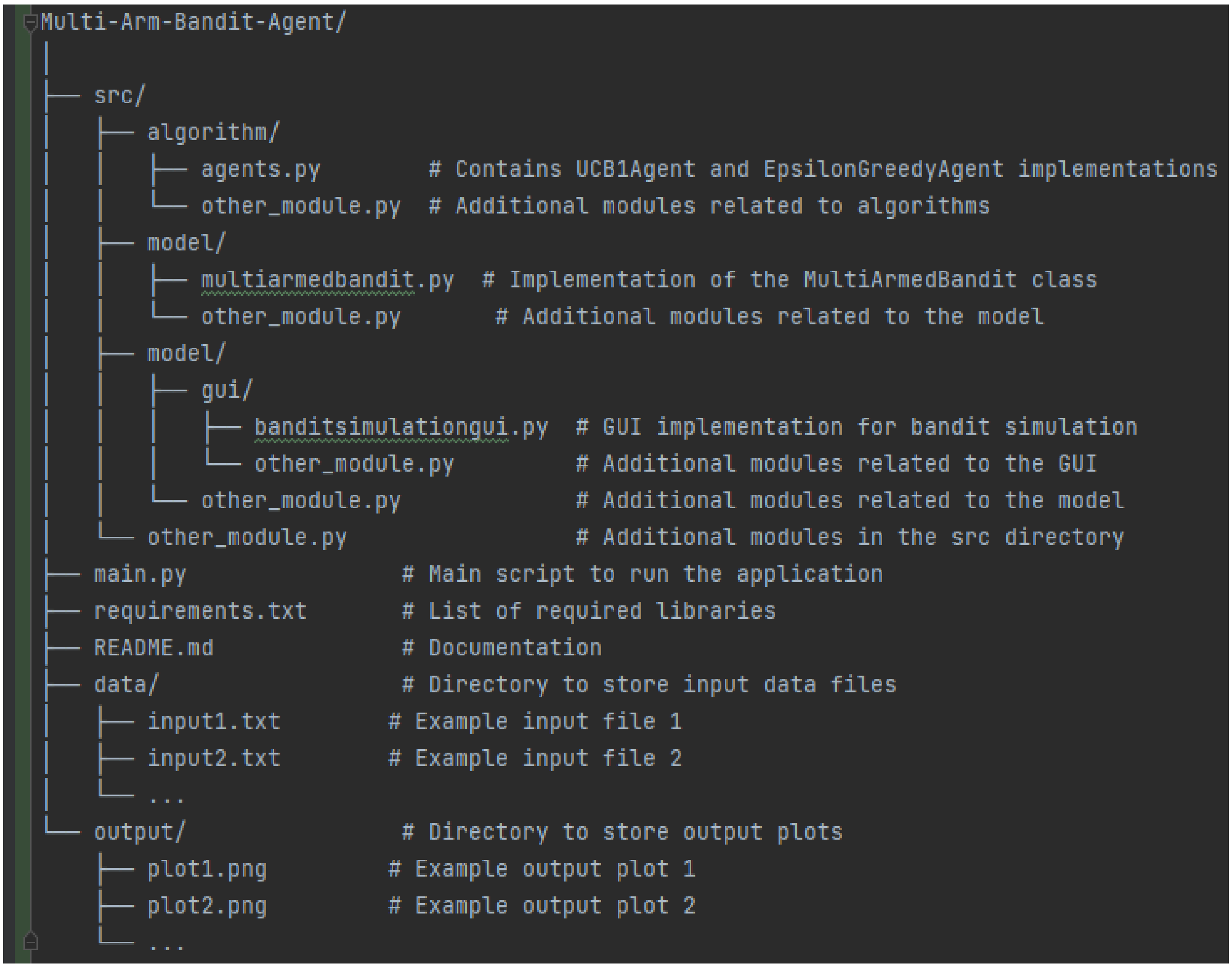


Output Structure

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
    └── ...
```

Additional 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.

! NOTE !

Ensure that any dependencies or required files are present in the same directory as the executable.

If there are any issues running the executable, check for error messages in the terminal or logs and troubleshoot accordingly.

Using the executable provides a more straightforward way for users to run the application without needing to directly interact with the Python scripts.