**SI 507 Final Checkpoint**

**Name Yujie Li**

**Project Code:**

<https://github.com/VectorLambda/Final_Project.git>

Readme.md is also uploaded to github repository. It is not displayed in the project document due to its length.

Required python packages to run this code: json, random, pandas, requests, and re(the regex package).

**Data Source:**

We used the Open Movie Database as the source of our data. The link to the database API is listed below:

<https://www.omdbapi.com/>

To extract data from the web API, we used IMDb ID to extract the data from API. We also provided a csv file that listed the names of movies with its corresponding IMDb ID. The cached data from web API is saved as a JSON file in movies.json.

The amount of data in the csv is about 100 movies entries. The limited sized is due to the API call requirement set by Open Movie Database (No more than 1000 calls daily). For each entry of data extracted from the API, there contains about 20 keys of information stored as a JSON object. For the purpose of the project, we only need to extract five fields of information from each object.(Titles, Rating, Genre, Plot, and Poster Link)

**Data Structure:**

The data structure we used for this project is a binary tree structure. Similar to the tree structure demonstrated in 20 Questions, we store a series of question for users to answer, based on their inputs, we will then navigate throughout the tree structure until it reaches a leaf node. For this project, instead of storing a string representation of question, we used a list to store all movies objects that meets the condition of the tree branch.

The code file we used to store the data in tree structure is listed in the code snippet below. In addition, we also have tree.py file, which the main code can call this function to print the tree structure in a more interpretable approach.

Below is a screenshot of what the tree structure looks like.

**Interaction and Presentation Plans:**

For the interaction of the code, we use a command terminal to display the performance of the code. The program with provides users with a series of questions on what genre of movie they may be looking for. If the user answers “yes” to any of the question, the program will return the primary information of at most five random movies (title, rating, plot, and link to poster). After displaying the result, the system will ask if the user is interested in the result, or that they want to refresh the display for a new set of recommendation. If the result is empty, that means the user’s request is not within the data listed in the data structure, and they should try again to search other categories.

Demo Video: