Flask Mockup 2

What is our GOAL for this MODULE?

The goal of this module is to complete our Flask API so we can start coding the React Native part.

What did we ACHIEVE in the class TODAY?

- Completed the Flask API
- Created classifiers and functions for movie recommendation

Which CONCEPTS/CODING BLOCKS did we cover today?

- Flask APIs
- Postman

How did we DO the activities?

1. On noticing, we observe that our data does not have the movie's poster link. Download the CSV from here containing the movie's poster links:

   https://raw.githubusercontent.com/whitehatjr/c-142/main/movie_links.csv

2. Move this CSV **movie_links.csv** to the Flask APIs directory and observe the CSV. It has 2 columns - **name and imdb_link.**
3. Create a new file **merge_csv.py** and in this file, make the imports and read our original CSV **movies.csv** here, that contains all the data of all the movies.

```
import csv

with open('movies.csv') as f:
   reader = csv.reader(f)
   data = list(reader)
   all_movies = data[1:]
   headers = data[0]
```

4. Let's add a new header and save these headers in a new csv **final.csv** as an append method:

```
headers.append("poster_link")

with open("final.csv", "a+") as f:
   csvwriter = csv.writer(f)
   csvwriter.writerow(headers)
```

5. Read the contents of the movie's poster link from the CSV you downloaded:

```
with open("movie_links.csv") as f:
   reader = csv.reader(f)
   data = list(reader)
   all_movie_links = data[1:]
```

6. Add the logic to merge the two data points (the original movie data with the movie's poster link) together. Note that the original data has 4,807 rows while our movie's poster data csv has 4,748 rows. This means that we do not have poster links to a few movies. Be mindful of that while writing your logic:

```
for movie_item in all_movies:
   poster_found = any(movie_item[8] in movie_link_items for movie_link_items in
all_movie_links)
   if poster_found:
       for movie_link_item in all_movie_links:
           if movie_item[8] == movie_link_item[0]:
               movie_item.append(movie_link_item[1])
               if len(movie_item) == 28:
                   with open("final.csv", "a+") as f:
                       csvwriter = csv.writer(f)
                       csvwriter.writerow(movie_item)
```

7. Now, we will use this **final.csv** instead of **movies.csv** in our Flask API.
8. Now if we talk about our second screen, we can keep 2 tabs in it:
    - **Tab 1 -** For movie recommendations based on what the user liked
    - **Tab 2 -** For giving the most popular movies based on demographic filtering
9. Create a new file **demographic_filtering.py**. Read the final.csv here into a DataFrame. Calculate the values of C, m and find the movies that have more votes than 0.9 quantile of the movie, just like how we did in Google Colab. Define a function to calculate the weighted rating and create a new column of score in the dataframe for all the movies. Create a variable with top 20 movies as a list.
10. Create a **GET API** to return the 20 most popular movies' data for our 2nd tab.
11. Now for the first tab, we create a new file **content_filtering.py**. Here we will do the same steps as what we did in Google Colab, but we don't want to do any sort of processing to this data, since we already downloaded the processed data from our Google Colab's df2. One thing that we need to ensure is that we want to drop the rows that do not have a valid metadata soup string. We can do that with the following code:

```
df = pd.read_csv('final.csv')
df = df[df['soup'].notna()]
```

12. Use the recommendation function that you created in **content_filtering.py** and make another **GET API** to return the list of recommended movies. Here, make sure that the same movie can be recommended for different movies. For example, Godfather 3 might get recommended for both Godfather and Godfather 2. Hence, we want to also ensure we take care of the duplicates in our data.
13. Test your API using Postman.

What’s NEXT?

In the next class, we will be working on the React Native Side UI Part of the App and build **Screen 1,** where the user will be able to tell if they like the movie, dislike the movie or did not watch.