

Movie Recommendation Application

Ryan S. Shaw

Northwest Missouri State University, Maryville MO 64468, USA
S546850@nwmissouri.edu

Abstract. Keywords: data analytics · movies · Python · web scraping

1 Introduction

I plan to work in the domain of web scraping. Primarily because it is the most interesting to me but also because I feel I did not get to fully explore web scraping in the web scraping course due to time constraints.

I would like to scrape my data from the online website IMDB primarily. I may also source data from Rotten Tomatoes.

I intend to solve the problem of trying to decide which movies to watch. I feel like this is something everyone struggles with at some point. Scrolling through endless movies on Netflix or some other streaming service and not knowing what to choose.

Steps taken would be as follows:

1. Scrape the data from the previously mentioned websites.
2. Clean/organize the data into a usable format.
3. Create a Python script that requests user information (name, birthdate, gender, favorite genre).
4. Create a model that can recommend some movies that the user may like based on their information and the movie rating.

Python will be a huge component for me in this project. I will need it to for just about every aspect of this project.

2 Data Sourcing

I used Python and web scraping (specifically the BeautifulSoup Module) to gather my data from the IMDB website. The data is in HTML format being written initially for a webpage and collected using Python. The attributes that will be gathered are: Title, Release Year, Rating, Runtime, Genre, Metascore, Movie Description and Votes. IMDB organizes movies by genre, so moving forward I would like to implement a method for users to search movie recommendations by genre.


```
movie_data.append(data)

with open(r'C:\Users\User\Desktop\Capstone\Module 3\ module3.csv', 'w', newline='') as csv_file:
    for y in movie_data:
        writer = csv.DictWriter(csv_file, fieldnames=y.keys())
        writer.writerow(y)
```

Fig. 3. New code to write to CSV file.

I changed the Writer method to DictWriter since it is specifically made to handle writing dictionary objects to CSV which is what I am doing here. This keeps each movie record on a line instead of printing each attribute of every movie on a new line.

I also added the newline argument. This tells Python exactly how I want it to handle new lines. As you can see in Fig. 1, it was skipping a line after each record. It no longer does this thanks to the newline argument being specified.

These were the biggest changes made using Python. The rest I was able to easily accomplish using Microsoft Excel as you can see in Fig. 4. I added a title row which shows the names of each attribute, bolded. I also centered all the data and expanded the cells to see the data better. I also deleted the Year column completely, as it is not important for my project and IMDB includes this information in the movie titles, making this redundant. This left me with 8 attributes and had no missing values that needed cleaning.

Movie Title	Rating	Runtime	Genres	IMDB User Rating	Metascore Rating	Description	Number of Votes
1. Indiana Jones and the Dial of Destiny (2023)	PG-13	154 min	Action, Adventure	6.9	58 Metascore	Archaeologist Indiana Jones races against time	Votes: 71,063
2. Sound of Freedom (2023)	PG-13	131 min	Action, Biography, Drama	8.2	43 Metascore	The incredible true story of a former governm	Votes: 22,491
3. Spider-Man: Across the Spider-Verse (2023)	PG	140 min	Animation, Action, Adventure	8.9	86 Metascore	Miles Morales catapults across the Multiverse,	Votes: 182,518
4. Mission: Impossible - Dead Reckoning Part One (2023)	PG-13	163 min	Action, Adventure, Thriller	8.1	81 Metascore	Ethan Hunt and his IMF team must track down	Votes: 45,855
5. The Flash (2023)	PG-13	144 min	Action, Adventure, Fantasy	7.1	56 Metascore	Barry Allen uses his super speed to change the	Votes: 88,374
6. Nimona (2023)	PG	101 min	Animation, Action, Adventure	7.7	75 Metascore	When a knight in a futuristic medieval world is	Votes: 15,615
7. Extraction II (2023)	R	122 min	Action, Thriller	7.1	57 Metascore	After barely surviving his grievous wounds fro	Votes: 96,544
8. Guardians of the Galaxy Vol. 3	PG-13	150 min	Action, Adventure, Comedy	8.1	64 Metascore	Still reeling from the loss of Gamora, Peter Qu	Votes: 213,698

Fig. 4. New output CSV file after data cleaning.

This leaves me with the following attributes:

1. Movie Title (Dependent Variable): The title of the movies.
2. Rating: (Independent Variable): Movie rating (E.g. G, PG - 13, R, etc.).
3. Runtime: Length of movie.
4. Genres (Independent Variable): Movie genre according to IMDB.
5. IMDB User Rating (Independent Variable): Average rating (out of 10 possible) given by IMDB users.
6. Metascore Rating (Independent Variable): Critic rating (out of 100 possible) assigned by movie critics.
7. Description: Movie description.
8. Number of Votes: Number of votes by IMDB users for that movie.

4 Analysis

Exploratory data analysis (EDA), refers to the act of investigating your dataset in order to uncover patterns, test hypotheses or assumptions, and discover anomalies. This step is essential for any data analytics project as helps spot missing or incorrect data, determine the most important values, find patterns or anomalies, and prove/disprove hypotheses.

There are four primary data analysis techniques:

1. Univariate Non-Graphical: This technique only involves 1 variable and does not deal with relationships.
2. Univariate Graphical: Also deals with only 1 variable and utilizes graphical methods to provide a fuller picture of the data.

3. Multivariate Non-Graphical: Data has multiple variables. Shows the relationships between data using statistics or cross-tabulation.
4. Multivariate Graphical: Data has multiple variables. Uses graphics to show relationships between the data. (This is the technique I used in my own EDA.)

I came up with four questions that I wanted to answer through data exploration in Tableau. The first question I wanted to know was: Is/are there movie genre(s) that get rated more highly by critics?

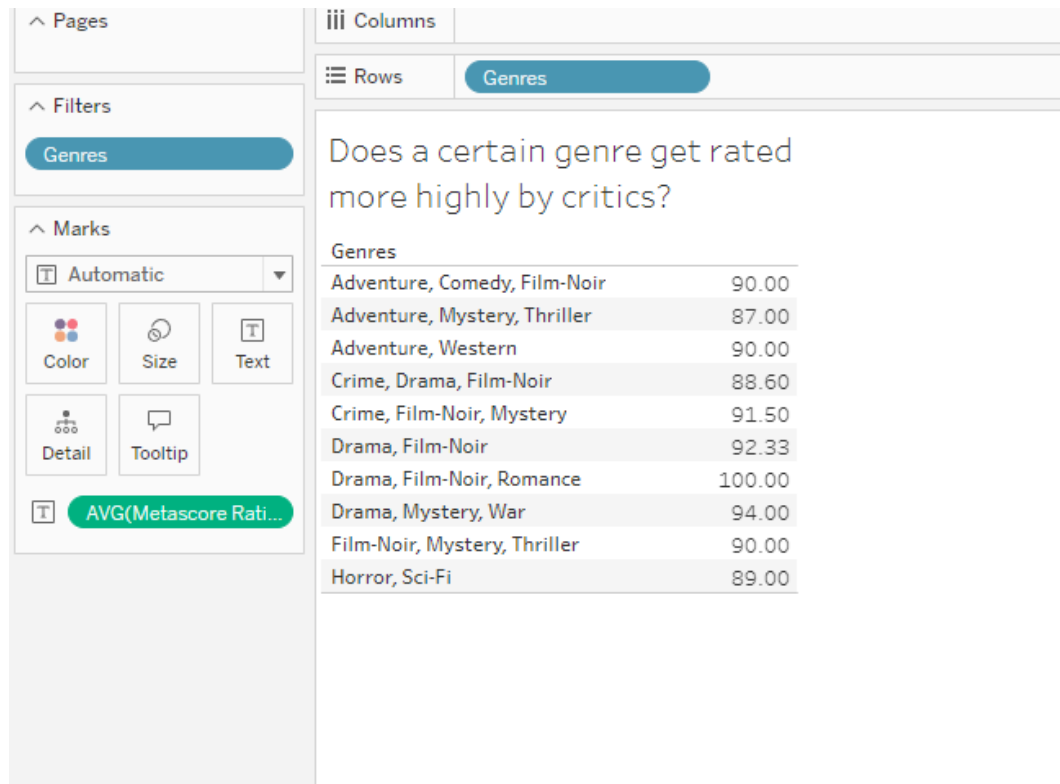


Fig. 5. Table showing the highest rated genres by critics.

As you can see in Figure 5, Drama, Adventure, Film-Noir, and Crime feature heavily in the top 10 highest rated genres by critic metascore on IMDB.

Next, is/are there movie genre(s) that get rated more highly by IMDB users?

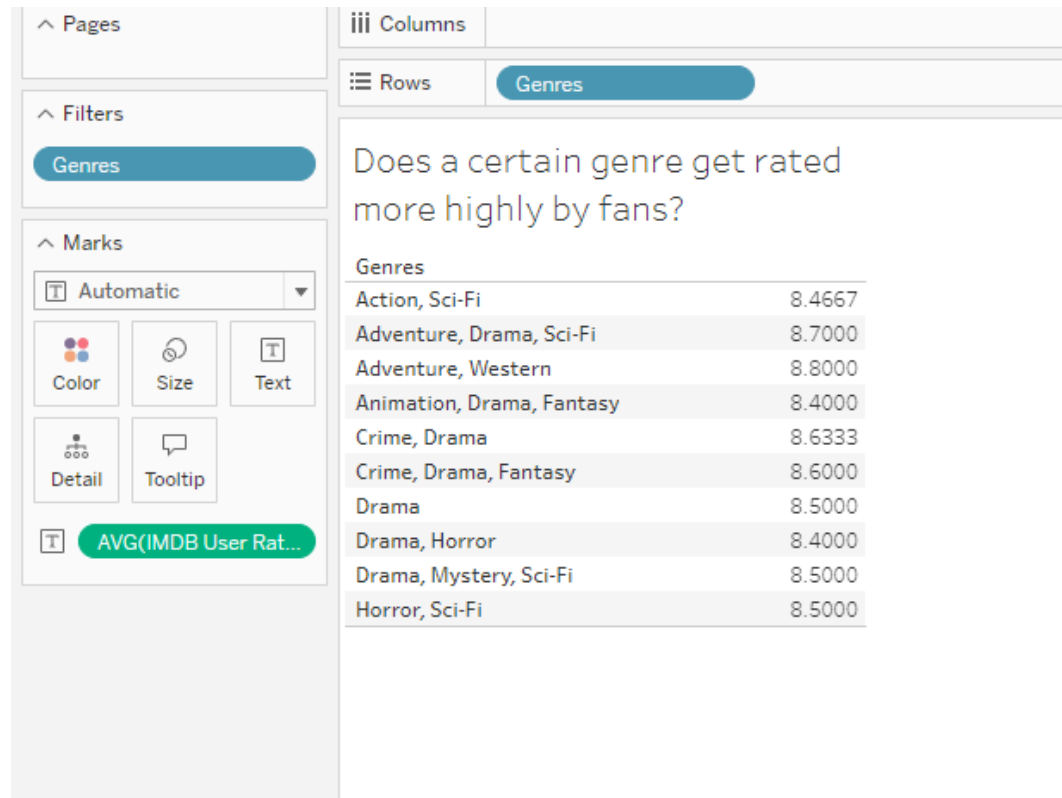


Fig. 6. Table showing the highest rated genres by IMDB users.

As shown in Figure 6, Drama features very heavily in the top 10 highest rated genres by IMDB user score, with Crime, Adventure, Fantasy, and Sci-Fi among some others, also being mentioned several times.

Are there any movies that show a large discrepancy in ratings between critics and fans?

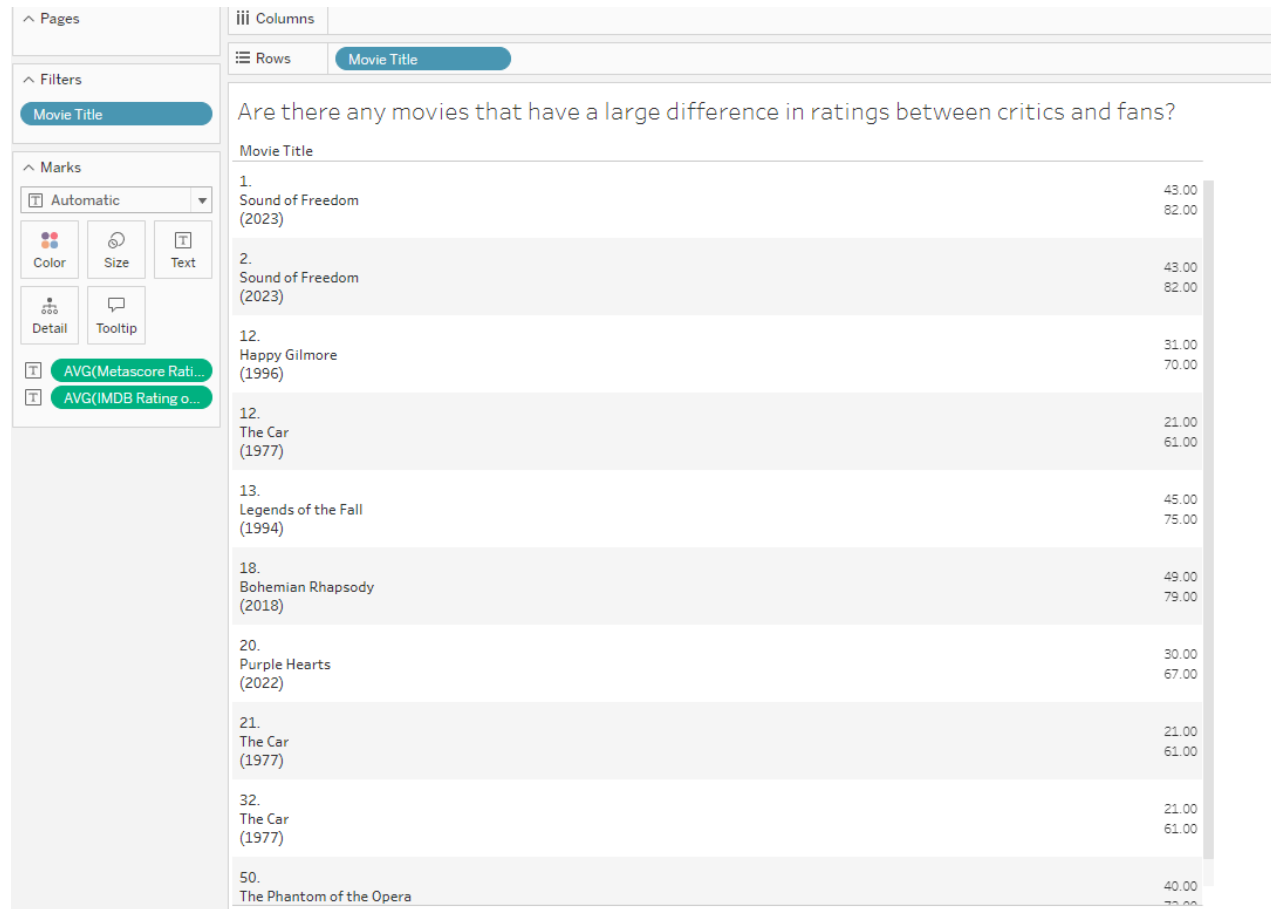


Fig. 7. Comparison of IMDB user rating vs critic rating.

Figure 7 shows the 10 movies with the largest difference in IMDB user rating and critic rating. Since IMDB user rating is only on a 10 point scale and metascore rating is on a 100 point scale, I first had to multiply all the user ratings by 10. This way I could make a more direct comparison between the two ratings. The top number on the right is the metascore rating given by critics and the bottom number is the IMDB user rating. This shows that casual movie watchers think quite highly of these films despite the poor critical perception.

Another thing to notice are the duplicate movie titles. This is because the movies are listed by genre and because a movie can belong to multiple genres, they can be listed several times. This is a good example of what can be revealed through proper EDA. In this case, I will need to further clean my data in order to remove any duplicate values.

Finally, does a certain movie rating (i.e., PG, PG - 13, R, etc.) get rated more highly than others?

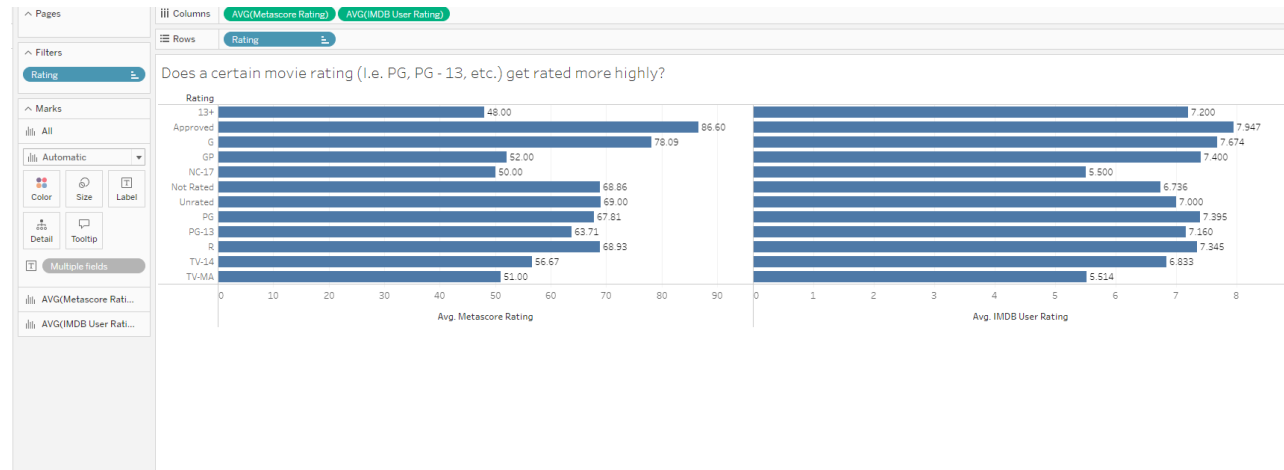


Fig. 8. Bar chart showing the average IMDB score for each movie rating.

Figure 8 shows that the highest rated film rating for both critics and IMDB users is "Approved". Approved is actually a rating given to movies prior to 1968 if they were deemed "moral". I suspect this data is an outlier as there would be much fewer movies produced before 1968 and even fewer still being watched today. Of those that are still being watched, they would have to be good in order to stand the test of time. The next highest would be G rated films.

5 Predictive Analysis

My ultimate goal is to provide an application that will recommend a movie to users based on already calculated scores on the IMDB website. For my case, predicting scores is not really necessary. However, I was curious to see if movie runtime had any effect on people's perception of the movies they watch and if I could use that to predict the final score.

To do this, I used a linear regression model in Microsoft Excel's data analysis add-in. The linear regression model attempts to establish the linear relationship between two variables (movie runtime and IMDB user rating in this case) based on a line of best fit. You can see the results of this model in Figure 9:

SUMMARY OUTPUT									
Regression Statistics									
Multiple R	0.328561153								
R Square	0.107952431								
Adjusted R Square	0.106855202								
Standard Error	0.817359589								
Observations	815								
ANOVA									
	<i>df</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>Significance F</i>				
Regression	1	65.72964454	65.72964454	98.38637501	5.73555E-22				
Residual	813	543.1463555	0.668076698						
Total	814	608.876							
	<i>Coefficients</i>	<i>Standard Error</i>	<i>t Stat</i>	<i>P-value</i>	<i>Lower 95%</i>	<i>Upper 95%</i>	<i>Lower 95.0%</i>	<i>Upper 95.0%</i>	
Intercept	5.955761455	0.142461047	41.80624525	9.5316E-205	5.676126635	6.235396275	5.676126635	6.235396275	
Runtime	0.01111526	0.001120604	9.918990625	5.73555E-22	0.008915642	0.013314878	0.008915642	0.013314878	

Fig. 9. Linear regression analysis results.

The Adjusted R-Squared value of 10.7 percent shows how good of a fit the linear line for this model is. This is not a good value and shows that runtime is not a good indicator of IMDB score. However, looking at other values going off of this model, the intercept value is 5.96 and the movie runtime coefficient is 0.0111. Using these values to try to make a prediction of a movie that was 120 minutes long would look like:

$$120 * 0.0111 + 5.96 = 7.29$$

This means, based on this model, a movie that is 120 minutes long would be rated at 7.29 out of 10.

6 Results

In Figure 10, we can see the opening portion of my code:

```

1 import pandas as pd
2 import re
3
4
5 pd.options.display.max_colwidth = 1000
6
7 unwanted = '()'
8 numbers = r'[0-9]'
9
10 genres = ['western', 'thriller', 'horror', 'sci-fi', 'mystery', 'film-noir', 'romance', 'musical', 'music', 'drama', 'war', 'sport',
11 | | | 'history', 'crime', 'comedy', 'fantasy', 'biography', 'family', 'animation', 'adventure', 'action']
12

```

Fig. 10. Beginning code for the movie recommendation app.

These lines are mostly importing the Python modules I need and declaring variables that will be used later in the program, with one line devoted to setting my display options so that all the information I want to display can be easily viewed at the end of my program.

Figure 11 shows the next two lines of code:

```

df = pd.read_csv("C:/Users/User/Desktop/Capstone/Module 6/module3.csv", encoding='utf8')

df['New Rating'] = df['IMDB User Rating'] * 10 + df['Metascore Rating']

```

Fig. 11. Two lines of code reading in a CSV file and performing a calculation.

The first line reads in the CSV file that was created earlier during the web scraping process. The second line creates a new dataframe column called 'New Rating' and populates it with new values that are the results of the following calculation:

$$\text{New Rating} = \text{IMDB User Rating} * 10 + \text{Metascore Rating}$$

I wanted the program to consider both IMDB User Rating and Metascore Rating when deciding which movies to recommend. However, I couldn't simply add the scores together as IMDB User Rating is on a 10 point scale and Metascore Rating is on a 100 point scale. This is why I multiply the IMDB User Rating by 10.

Figure 12 shows the part of the code that asks the user questions and takes user input:

```

while True:
    favorite_genre = input('What genre movie do you enjoy watching?').title()
    if favorite_genre.lower() not in genres:
        print("I am sorry but we do not have a movie of that genre. Please choose another.")
        continue
    else:
        break

while True:
    try:
        age = int(input('How old is the youngest person that will be watching a movie with you?'))
    except ValueError:
        print("I am sorry but that is not a valid response. Please enter a whole number for the age.")
        continue
    else:
        break

if age < 17:
    while True:
        parent_present = input('Will the child\'s parent or guardian be present during the film?').lower()
        if parent_present not in ['yes', 'no']:
            print('I am sorry but you must enter "yes" or "no" as a valid response.')
            continue
        else:
            break

```

Fig. 12. Code that takes program user input.

The first question asks the user which genre they enjoy watching the most. The program not only takes user input, but also verifies that what the user input is a valid response by storing the response in a variable and comparing that variable to the 'genres' list that can be seen in Figure 10. Figure 13 shows what happens when a user enters an invalid response:

```

PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python311/python
What genre movie do you enjoy watching?fantasyy
I am sorry but we do not have a movie of that genre. Please choose another.
What genre movie do you enjoy watching?

```

Fig. 13. User entered an invalid response.

In this case, the user simply misspelled the genre. However, if they entered a genre that did not exist or any other invalid response, the program informs them that they must choose a different genre. It will continue to loop through this process until the user enters a valid genre.

The second question asked of the user that can be seen in Figure 12 is how old the youngest person watching the movie will be. This is important as the movies scraped from IMDB vary greatly in content and MPAA ratings and some may not be suitable for young children. Similar to the previous question, the program will verify the response is valid:

```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python311/python.exe "
What genre movie do you enjoy watching?fantasyy
I am sorry but we do not have a movie of that genre. Please choose another.
What genre movie do you enjoy watching?fantasy
How old is the youngest person that will be watching a movie with you?dog
I am sorry but that is not a valid response. Please enter a whole number for the age.
How old is the youngest person that will be watching a movie with you?21
```

Fig. 14. Program asking user age.

Figure 14 shows this in action. After responding with a valid genre, the program then asks the age question and the user enters a playful response of "dog" instead of an age. Just like in the previous question, the program will continue to loop until the user enters an integer for the age.

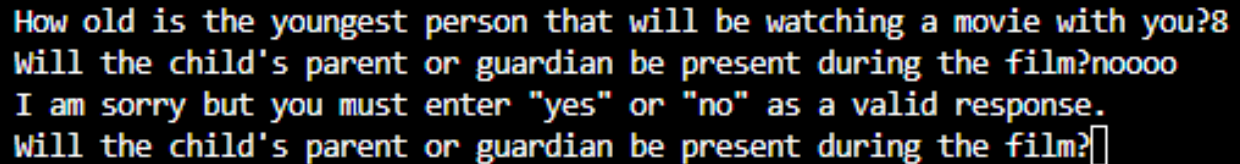
Figure 15 shows the final question asked of the user:

```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Python/Python311/p
What genre movie do you enjoy watching?thriller
How old is the youngest person that will be watching a movie with you?8
Will the child's parent or guardian be present during the film?
```

Fig. 15. Program asking if parent or guardian will be present.

This question is not always asked. If the response to the the age question is greater than 17, like in Figure 14, this question will be skipped. However, in

Figure 15 the user answered the age question with the number 8. In this case, with age being less than 17, the program asks the user if the child's parent or guardian will be present during the film. Once again, we verify the user's response and in this case, the only valid responses are "yes" or "no". We can see this in action in Figure 16:

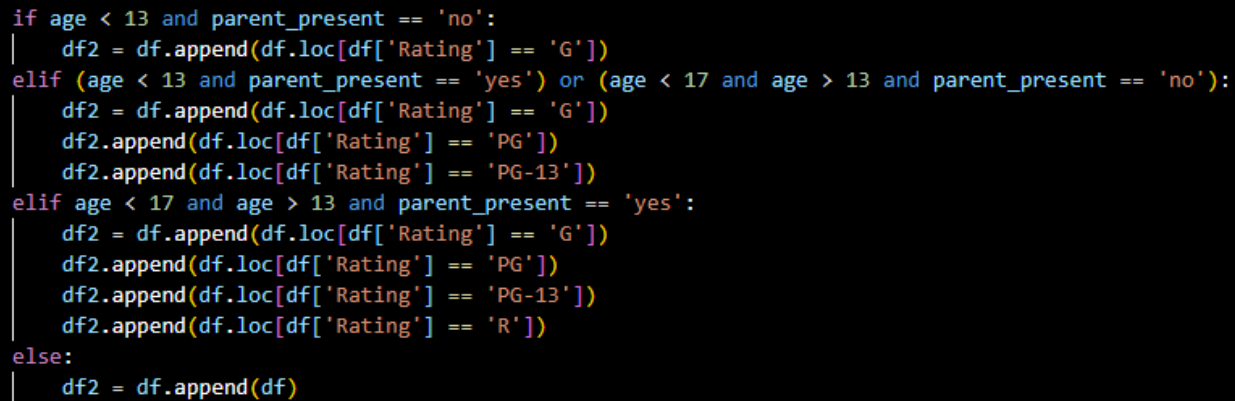


```
How old is the youngest person that will be watching a movie with you?8
Will the child's parent or guardian be present during the film?noooo
I am sorry but you must enter "yes" or "no" as a valid response.
Will the child's parent or guardian be present during the film?█
```

Fig. 16. Will the child's parent or guardian be present question.

In this case, the user held the "O" key too long and the program did not accept it.

Figure 17 shows the next lines of code:



```
if age < 13 and parent_present == 'no':
    df2 = df.append(df.loc[df['Rating'] == 'G'])
elif (age < 13 and parent_present == 'yes') or (age < 17 and age > 13 and parent_present == 'no'):
    df2 = df.append(df.loc[df['Rating'] == 'G'])
    df2.append(df.loc[df['Rating'] == 'PG'])
    df2.append(df.loc[df['Rating'] == 'PG-13'])
elif age < 17 and age > 13 and parent_present == 'yes':
    df2 = df.append(df.loc[df['Rating'] == 'G'])
    df2.append(df.loc[df['Rating'] == 'PG'])
    df2.append(df.loc[df['Rating'] == 'PG-13'])
    df2.append(df.loc[df['Rating'] == 'R'])
else:
    df2 = df.append(df)
```

Fig. 17. Code searching for movies based on user input.

These lines of code create a new dataframe (df2 in this case) and populates it with movie information based on the questions asked previously. Depending on the user's answer to the youngest person's age question and if their parent or guardian will be present, the program will populate df2 with values from df that have 'Rating' equal to G, PG, PG - 13, or R.

Next, strip our movie down further with the next lines of code:

```

df3 = df2[df2['Genres'].str.contains(favorite_genre)]

df4 = df3.loc[df3['New Rating'].idxmax()]

result = df4['Movie Title']
result_summary = df4['Description']

```

Fig. 18. Code performing final movie search.

Figure 18 shows us creating two more dataframes, df3 is the result of searching df2 for all the movies with a 'Genres' column equal to the user's favorite genre which they input earlier in the program and df4 is all the movie information for the movie in df3 which has the highest 'New Rating' value. Then, we store only the values from the 'Movie Title' and 'Description' columns in the df4 dataframe in the variables 'result' and 'result_summary'.

Finally, we get to our last line of code:

```

print('Based on your genre preference and the audience, this is your movie recommendation ' + result + ' Here is a brief summary: ' + result_summary)

```

Fig. 19. Print code showing the user results.

Figure 19 shows the print statement that informs the user of their movie recommendation. We can see this in Figure 20 where the program recommended "The Third Man":

```

df2.append(df.loc[df['Rating'] == 'PG-13'])
Based on your genre preference and the audience, this is your movie recommendation .
The Third Man
() Here is a brief summary: Pulp novelist Holly Martins travels to shadowy, postwar Vienna, only to find himself investigating the mysterious death of an old friend, Harry Lime.
PS C:\Users\User> 

```

Fig. 20. Final movie recommendation results.

In this case, the user's favorite genre was Thriller, the youngest person watching was 8 years old, and their parents were going to be present during the film. The program picked a PG - 13 rated movie and displayed the results to the user.

7 Limitations

8 Conclusion

□

References

1. Imdb: Ratings, reviews, and where to watch the best movies and tv, <https://www.imdb.com/>
2. Rotten tomatoes: Movies — tv shows, <https://www.rottentomatoes.com/>