

Movie Recommender System

July 2023 - Arman Kazemi 98222078



Overview

Movie recommendation systems are an integral part of the entertainment industry, aiming to provide personalized movie suggestions to users based on their preferences. The project entails analyzing a large dataset of movies, incorporating user preferences, and implementing algorithms to generate relevant movie recommendations.

Dataset

The Movies Dataset is a collection of data on 45,000 movies and contains information such as budget, revenue, release dates, languages, production countries, and companies for each movie. The dataset also includes ratings of 270,000 users for these movies. The data was obtained from The Movie Database (TMDb) API.

Method

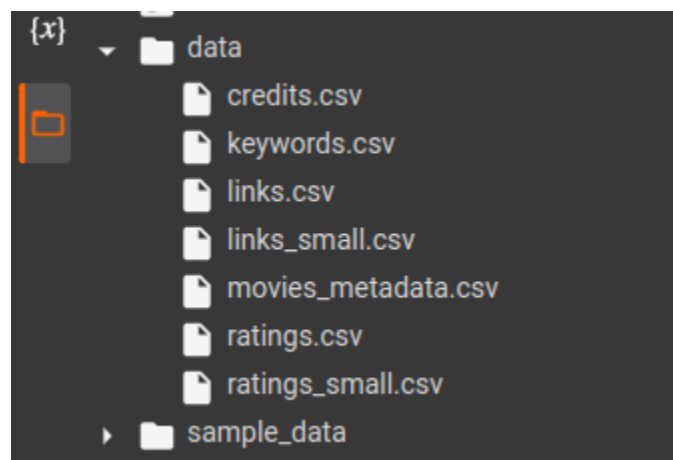
In this project, A few recommendation algorithms were implemented (content-based, popularity-based, and collaborative filtering) and built and ensembled to develop our final recommendation system. There were two MovieLens datasets. [The Full Dataset, The Small Dataset]

In the end, we used the **Hybrid** method to recommend base users.

Steps:

- **Loading data**

At first, the dataset was downloaded to the Colab notebook using Kaggle-token. Then it was unzipped.



And here we have the datasets, for example, *movies_metadata*:

Data shape: (45466, 24)

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_date	revenue	runtime	spoken_languages	status
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...}	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his room.	...	1995-10-30	373554033.0	81.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}]	NaN	844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an enchanted board game that opens the door to a magical world of adventure, the siblings discover an enchanted board game that opens the door to a magical world of adventure.	...	1995-12-15	262797249.0	104.0	[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'es', 'name': 'Spanish'}]	Released
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collection', ...}	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud between two families.	...	1995-12-22	0.0	101.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}]	NaN	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the women of the film find their way to the top.	...	1995-12-22	81452156.0	127.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released
4	False	{'id': 96871, 'name': 'Father of the Bride Collection', ...}	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11662	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his first wedding, he finds out that his son is getting married again.	...	1995-02-10	76578911.0	106.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released

5 rows x 24 columns

The shape of the dataset: (45466, 24)

- **Data cleaning**

Raw data usually has null values or wrong inputs. So before any action, data cleaning must be done. So, in the beginning, the null values of the dataset were observed.

```
Number of null values of columns:
```

```
adult          0
belongs_to_collection  40972
budget         0
genres         0
homepage       37684
id             0
imdb_id        17
original_language  11
original_title  0
overview       954
popularity     5
poster_path    386
production_companies  3
production_countries  3
release_date   87
revenue        6
runtime        263
spoken_languages  6
status         87
tagline        25054
title          6
video          6
vote_average   6
vote_count     6
```

```
Useless features: ['belongs_to_collection', 'homepage']
```

The columns with more than 70% of null values were pointed out.

I checked which columns have more than 70% null value. (`belongs_to_collection`, `homepage`)

After that, two useless columns were removed (**`belongs_to_collection`**, **`homepage`**), then the null values were filled by appropriate inputs according to their types.

```
adult          object
budget         object
genres         object
id             object
imdb_id        object
original_language object
original_title object
overview       object
popularity     object
poster_path    object
production_companies object
production_countries object
release_date   object
revenue        float64
runtime        float64
spoken_languages object
status         object
tagline        object
title          object
video          object
vote_average   float64
vote_count     float64
dtype: object
```

So I started to fill columns:

- Popularity: With the mean of not null values of that column
- revenue: With the mean of not null values of that column
- runtime: With the mean of not null values of that column
- vote_average: With the mean of not null values of that column
- vote_count: With the mean of not null values of that column
- Imdb_id: With random string-id
- original_language: 'en'
- overview: "",
- poster_path: '/random.jpg',
- production_companies: "[{'name': 'Warner Bros.', 'id': 6194}]",
- production_countries: "[{'iso_3166_1': 'US', 'name': 'United States of America'}]",
- release_date: With a random date in range of not value of the column,
- spoken_languages: "[{'iso_639_1': 'en', 'name': 'English'}]",
- status: 'Released',
- title: 'Nul',
- video: False,

```

adult      0
budget     0
genres     0
id         0
imdb_id    0
original_language  0
original_title  0
overview   0
popularity 0
poster_path      0
production_companies  0
production_countries  0
release_date      0
revenue           0
runtime           0
spoken_languages  0
status           0
tagline          25054
title            0
video            0
vote_average     0
vote_count       0
dtype: int64

```

Can not fill tagline (We want to save what movies really have tagline)

After all, the *id* column was cased to Integer type, but it threw an error:

```

# cast to integer
data['id'] = data['id'].astype('int')
-----
ValueError                                Traceback (most recent call last)
<ipython-input-187-e377d143c1c5> in <cell line: 2>()
      1 # cast to integer
----> 2 data['id'] = data['id'].astype('int')

----- 6 frames -----
/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in
astype_nansafe(arr, dtype, copy, skipna)
    168     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
    169         # Explicit copy, or required since NumPy can't view from / to object.
--> 170         return arr.astype(dtype, copy=True)
    171
    172     return arr.astype(dtype, copy=copy)

ValueError: invalid literal for int() with base 10: '1997-08-20'

```

```
[ ] data[data['id'] == '1997-08-20']
```

adult		budget	genres	id	imdb_id	original_language	original_title	overview	p
19730	Written by Ørnås	/ff9qCepilowshEtG2GYWwzt2bs4.jpg	[{'name': 'Carousel Productions', 'id': 11176}]...	1997-08-20	0	104.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released	

1 rows × 22 columns

The id of one of the rows was '1997-08-20', so it was found and removed. And then I tried again to do the casting, but there was still an error. In the end, three rows needed to be removed: **[19730, 29503, 35587]**

Finally, our data was cleaned.

- **Use IMDB's weighted rating formula**

I used the IMDB's weighted rating formula with the TMDB rating to construct my chart.

The formula:

$$\left(\frac{v}{v+m} \cdot R \right) + \left(\frac{m}{v+m} \cdot C \right)$$

- v is the number of votes for the movie
- m is the minimum number of votes required to be listed in the chart
- R is the average rating of the movie
- C is the mean vote across the whole report

In the next step, an appropriate value for m was determined, and **the minimum votes required to be listed in the chart**. For example, I used the 95th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 95% of the movies in the list.

So for this goal, I extracted the **year** from the release-date column and selected some columns from data for our qualified dataset. (Which are in 95%):

m: 433.9, C: 5.245

Therefore, a movie must have at least 433.9 votes on TMDb to qualify for the chart. I also saw that the average rating for a movie on IMDB is 5.245 on a scale of 1 to 10.

	title	year	vote_count	vote_average	popularity	genres
0	Toy Story	1995	5415	7	21.946943	['id': 16, 'name': 'Animation'], ['id': 35, 'name': 'Comedy'], ['id': 10428, 'name': 'Family']
1	Jumanji	1995	2413	6	17.015539	['id': 12, 'name': 'Adventure'], ['id': 14, 'name': 'Fantasy'], ['id': 10428, 'name': 'Family']
5	Heat	1995	1886	7	17.924927	['id': 28, 'name': 'Action'], ['id': 80, 'name': 'Crime'], ['id': 10428, 'name': 'Family']
9	GoldenEye	1995	1194	6	14.686036	['id': 12, 'name': 'Adventure'], ['id': 28, 'name': 'Action'], ['id': 10428, 'name': 'Family']
15	Casino	1995	1343	7	10.137389	['id': 18, 'name': 'Drama'], ['id': 80, 'name': 'Crime'], ['id': 10428, 'name': 'Family']
...
44624	What Happened to Monday	2017	598	7	3.393351	['id': 878, 'name': 'Science Fiction'], ['id': 10428, 'name': 'Family']
44632	Atomic Blonde	2017	748	6	3.393351	['id': 28, 'name': 'Action'], ['id': 53, 'name': 'Thriller'], ['id': 10428, 'name': 'Family']
44678	Dunkirk	2017	2712	7	3.393351	['id': 28, 'name': 'Action'], ['id': 18, 'name': 'Drama'], ['id': 10428, 'name': 'Family']
44842	Transformers: The Last Knight	2017	1440	6	3.393351	['id': 28, 'name': 'Action'], ['id': 878, 'name': 'Science Fiction'], ['id': 10428, 'name': 'Family']
45014	The Dark Tower	2017	688	5	3.393351	['id': 28, 'name': 'Action'], ['id': 37, 'name': 'Horror'], ['id': 10428, 'name': 'Family']

2274 rows x 6 columns

The shape of data: (2274, 6)

2274 Movies qualified to be on our chart.

Then weighted_rate base TMDb formula was added. So now we can find the top 250 movies:

	title	year	vote_count	vote_average	popularity	genres	weighted_rating
15480	Inception	2010	14075	8	29.108149	['id': 28, 'name': 'Action'], ['id': 53, 'name': 'Thriller'], ['id': 10428, 'name': 'Family']	7.917606
12481	The Dark Knight	2008	12269	8	123.167259	['id': 18, 'name': 'Drama'], ['id': 28, 'name': 'Action'], ['id': 10428, 'name': 'Family']	7.905892
22879	Interstellar	2014	11187	8	32.213481	['id': 12, 'name': 'Adventure'], ['id': 18, 'name': 'Drama'], ['id': 10428, 'name': 'Family']	7.897130
2843	Fight Club	1999	9678	8	63.869599	['id': 18, 'name': 'Drama'], ['id': 10428, 'name': 'Family']	7.881778
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	['id': 12, 'name': 'Adventure'], ['id': 14, 'name': 'Fantasy'], ['id': 10428, 'name': 'Family']	7.871814
292	Pulp Fiction	1994	8670	8	140.950236	['id': 53, 'name': 'Thriller'], ['id': 80, 'name': 'Crime'], ['id': 10428, 'name': 'Family']	7.868689
314	The Shawshank Redemption	1994	8358	8	51.645403	['id': 18, 'name': 'Drama'], ['id': 80, 'name': 'Crime'], ['id': 10428, 'name': 'Family']	7.864029
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	['id': 12, 'name': 'Adventure'], ['id': 14, 'name': 'Fantasy'], ['id': 10428, 'name': 'Family']	7.861956
351	Forrest Gump	1994	8147	8	48.307194	['id': 35, 'name': 'Comedy'], ['id': 18, 'name': 'Drama'], ['id': 10428, 'name': 'Family']	7.860685
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	['id': 12, 'name': 'Adventure'], ['id': 14, 'name': 'Fantasy'], ['id': 10428, 'name': 'Family']	7.851955
256	Star Wars	1977	6778	8	42.149697	['id': 12, 'name': 'Adventure'], ['id': 28, 'name': 'Action'], ['id': 10428, 'name': 'Family']	7.834240
1225	Back to the Future	1985	6239	8	25.778509	['id': 12, 'name': 'Adventure'], ['id': 35, 'name': 'Comedy'], ['id': 10428, 'name': 'Family']	7.820851
834	The Godfather	1972	6024	8	41.109264	['id': 18, 'name': 'Drama'], ['id': 80, 'name': 'Crime'], ['id': 10428, 'name': 'Family']	7.814886
1154	The Empire Strikes Back	1980	5998	8	19.470959	['id': 12, 'name': 'Adventure'], ['id': 28, 'name': 'Action'], ['id': 10428, 'name': 'Family']	7.814138
46	Se7en	1995	5915	8	18.457430	['id': 80, 'name': 'Crime'], ['id': 9648, 'name': 'Mystery'], ['id': 10428, 'name': 'Family']	7.811708

The top 15 movies.

- **Build Content-Based Recommenders based on the *genre* (for a start)**

In the previous part, we saw the three Films, *Inception*, *The Dark Knight*, and *Interstellar*, take place at the very top of our chart. The chart also indicates a strong bias of TMDb Users towards particular genres and directors.

Then I constructed our function that builds charts for particular genres. I relaxed our default conditions to the 85th percentile instead of 95.

For that, I cast the genres column that contains an array of names of movie genres instead of the genres object.

	adult	budget	genres	id	imdb_id	original_language	original_title	overview	popularity	poster_path
0	False	30000000	[Animation, Comedy, Family]	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhlRbceoE9IR4veEXuwCC2wARTG.jpg
1	False	65000000	[Adventure, Fantasy, Family]	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	17.015539	/vzmL6iP7aPKNKPRTFnZmiUfcyV.jpg
2	False	0	[Romance, Comedy]	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	11.712900	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg
3	False	16000000	[Comedy, Drama, Romance]	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	3.859495	/16XOMpEaLWkrPqSQqhTmeJuqQL.jpg
4	False	0	[Comedy]	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	8.387519	/e64sOI48hQXyru7naBFyssKFxVd.jpg
...
45461	False	0	[Drama, Family]	439050	tt6209470	fa	رگ و خرب	Rising and falling between a man and woman.	3.393351	/jldsY1lnId4tTWPx8es3uzsB1t8.jpg

So, for example, top movies in the **Romance** genre are:

	title	year	vote_count	vote_average	popularity	weighted_rating
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.565285
351	Forrest Gump	1994	8147	8	48.307194	7.971357
876	Vertigo	1958	1162	8	18.208220	7.811667
40251	Your Name.	2016	1030	8	3.393351	7.789489
883	Some Like It Hot	1959	835	8	11.845107	7.745154
1132	Cinema Paradiso	1988	834	8	14.177005	7.744878
19901	Paperman	2012	734	8	7.198633	7.713951
37863	Sing Street	2016	669	8	3.393351	7.689483
882	The Apartment	1960	498	8	11.994281	7.599317
38718	The Handmaiden	2016	453	8	3.393351	7.566166
3189	City Lights	1931	444	8	10.891524	7.558867
24886	The Way He Looks	2014	262	8	5.711274	7.331363
45437	In a Heartbeat	2017	146	8	3.393351	7.003959
1639	Titanic	1997	7770	7	26.889070	6.981546
19731	Silver Linings Playbook	2012	4840	7	14.488111	6.970581

- **Build Content-Based Recommenders based on *Movie Overviews and Taglines***

To achieve this goal, I loaded the **links_small** dataset to do this BC, just small movies have taglines.

The shape of links_small: (9099, 23)

So I tried to do this step based on the **overview and tagline** using the **TfidfVectorizer** library to calculate **Cosine Similarity**.

The TfidfVectorizer class in scikit-learn is used to convert a collection of raw documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents.

The Cosine Similarity was used to calculate a numeric quantity that denotes the similarity between two movies:

$$\frac{x \cdot y^T}{||x|| \cdot ||y||}$$

When the TF-IDF Vectorizer was used, calculating the Dot Product directly gave the Cosine Similarity Score. Therefore, I used sklearn's *linear_kernel* instead of *cosine_similarities* since it is much faster.

```
array([1.          , 0.00680476, 0.          , ..., 0.          , 0.00344913,
       0.          ])
```

For example, `get_recommendations('The Dark Knight,' 15)`:

```
7931          The Dark Knight Rises
132           Batman Forever
1113          Batman Returns
8227  Batman: The Dark Knight Returns, Part 2
7565          Batman: Under the Red Hood
524           Batman
7901          Batman: Year One
2579  Batman: Mask of the Phantasm
2696           JFK
8165  Batman: The Dark Knight Returns, Part 1
6144          Batman Begins
7933  Sherlock Holmes: A Game of Shadows
5511           To End All Wars
4489           Q & A
7344  Law Abiding Citizen
Name: title, dtype: object
```

The top movie consists of 'The Dark Knight' based on description and tagline

We see that for The Dark Knight, our system is able to identify it as a Batman film and subsequently recommend other Batman films as its top recommendations.

- **Build Content-Based Recommenders based on *Movie Cast, Crew, Keywords, and Genre***

I loaded *credits and keywords* from CSV:

	cast	crew	id
0	[{'cast_id': 14, 'character': 'Woody (voice)', ...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...	862
1	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...	8844
2	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...	15602
3	[{'cast_id': 1, 'character': 'Savannah Vannah...	[{'credit_id': '52fe44779251416c91011acb', 'de...	31357
4	[{'cast_id': 1, 'character': 'George Banks', '...	[{'credit_id': '52fe44959251416c75039ed7', 'de...	11862

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290, ...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...
2	15602	[{'id': 1495, 'name': 'fishing'}, {'id': 12392...
3	31357	[{'id': 818, 'name': 'based on novel'}, {'id': ...
4	11862	[{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n...

I merged these two tables to the data table base id and filter them based on *links_small*. The shape of data: (9219, 26)

Now we have our cast, crew, genres, and credits all in one data frame. Then I wrangled this a little more using the following intuitions:

- Crew: From the crew, we only picked the director as our feature since the others don't contribute that much to the movie's feel.
- Cast: Choosing Cast is a little more tricky. Lesser-known actors and minor roles do not really affect people's opinion of a movie.

Therefore, we must only select the significant characters and their respective actors. Arbitrarily we chose the top 3 actors that appear in the credits list.

For example, `get_recommendations('The Dark Knight', 15):`

```
8031          The Dark Knight Rises
6218          Batman Begins
6623          The Prestige
2085          Following
7648          Inception
4145          Insomnia
3381          Memento
8613          Interstellar
7659          Batman: Under the Red Hood
1134          Batman Returns
8927          Kidnapping Mr. Heineken
5943          Thursday
1260          Batman & Robin
9024          Batman v Superman: Dawn of Justice
4021          The Long Good Friday
Name: title, dtype: object
```

- **Final Content-Based Recommendations system**

One thing we notice about our recommendation system is that it recommends movies regardless of ratings and popularity.

Therefore, we added a mechanism to remove bad movies and return popular movies with an excellent critical response.

I took the top 25 movies based on similarity scores and calculated the vote of the 60th percentile movie. Then, using this as the value of m , I calculated the weighted rating of each movie using IMDB's formula as we did in the Simple Recommender section.

	title	vote_count	vote_average	year	wr
7648	Inception	14075	8	2010	7.917606
8613	Interstellar	11187	8	2014	7.897130
6623	The Prestige	4510	8	2006	7.758198
3381	Memento	4168	8	2000	7.740228
8031	The Dark Knight Rises	9263	7	2012	6.921465
6218	Batman Begins	7511	7	2005	6.904147
1134	Batman Returns	1706	6	1992	5.846887
132	Batman Forever	1529	5	1995	5.054131
9024	Batman v Superman: Dawn of Justice	7189	5	2016	5.013939
1260	Batman & Robin	1447	4	1997	4.287178

- Collaborative Filtering

Our content-based engine had limitations. It only suggested movies similar to a certain movie but couldn't capture user tastes across genres. It also lacked personalization, providing the same recommendations for everyone querying a movie. To overcome this, we used Collaborative Filtering with the Surprise library. It utilized powerful algorithms like SVD to minimize RMSE and provide accurate recommendations.

I loaded *ratings_small* data and did what was necessary.

Example of prediction: User: 1, Movie: 302, Result:

```
Prediction(uid=1, iid=302, r_ui=3, est=2.6549142760525566, details={'was_impossible': False})
```

- Hybrid Recommender

In that section, I attempted to build a simple hybrid recommender that combined techniques implemented in the content-based and collaborative filter-based engines. This is how it worked:

- Input: User ID and the Title of a Movie
- Output: Similar movies are sorted on the basis of expected ratings by that particular user.

Example:

User: 25, Movie: Avatar

	title	vote_count	vote_average	year	id	est
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.748513
974	Aliens	3282.0	7.7	1986	679	3.598249
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.558352
1011	The Terminator	4208.0	7.4	1984	218	3.498650
2014	Fantastic Planet	140.0	7.6	1973	16306	3.398614
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.335864
1376	Titanic	7770.0	7.5	1997	597	3.316582
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	3.305522
922	The Abyss	822.0	7.1	1989	2756	3.294811
4017	Hawk the Slayer	13.0	4.5	1980	25628	3.223319
3060	Sinbad and the Eye of the Tiger	39.0	6.3	1977	11940	3.177764
344	True Lies	1138.0	6.8	1994	36955	3.176582
831	Escape to Witch Mountain	60.0	6.5	1975	14821	3.159233

User: 80, Movie: Avatar

	title	vote_count	vote_average	year	id	est
974	Aliens	3282.0	7.7	1986	679	4.148646
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.939488
1011	The Terminator	4208.0	7.4	1984	218	3.768555
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.744055
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	3.743479
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.709726
2014	Fantastic Planet	140.0	7.6	1973	16306	3.614358
1376	Titanic	7770.0	7.5	1997	597	3.532304
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.512823
344	True Lies	1138.0	6.8	1994	36955	3.489947
3060	Sinbad and the Eye of the Tiger	39.0	6.3	1977	11940	3.473438
4347	Piranha Part Two: The Spawning	41.0	3.9	1981	31646	3.436256
831	Escape to Witch Mountain	60.0	6.5	1975	14821	3.410601

- **Make it compatible with Front-End**

An API file was written the help the model's search functions work with the clean and functioning dataset.

- Search function, search on movie titles and genres and return movie titles:

```
search('ro')  
[  
  'Cutthroat Island',  
  'Four Rooms',  
  'Dangerous Minds',  
  'Across the Sea of Time',  
  'Mighty Aphrodite',  
  'From Dusk Till Dawn',  
  'Bed of Roses',  
  'The Crossing Guard',  
  'The Juror',  
  'Vampire in Brooklyn',  
  'Broken Arrow',  
  'Bottle Rocket',  
  'Mr. Wrong',  
  'Rumble in the Bronx',  
]
```


- User search function: Get user id, get top movies which user rates and recommend base them using hybrid function.

```
user_search(3)
```

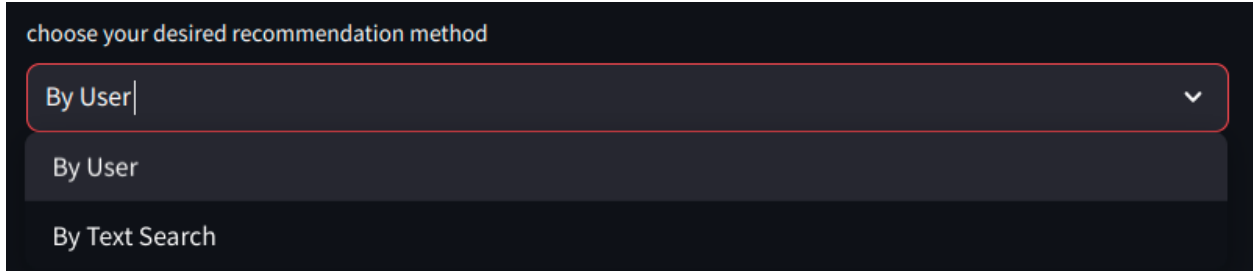
	title	vote_count	vote_average	year	id	est
21	The Matrix	9079.0	7.9	1999	603	3.969753
20	Ghost in the Shell	854.0	7.8	1995	9323	4.153080
47	Amélie	3403.0	7.8	2001	194	3.618942
42	Breathless	322.0	7.7	1960	269	3.724446
22	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.703610
31	Three Colors: Blue	311.0	7.7	1993	108	3.887627
37	Pierrot le Fou	134.0	7.7	1965	2786	3.517382
0	Paris, Texas	282.0	7.7	1984	655	3.995019
48	Pierrot le Fou	134.0	7.7	1965	2786	3.517382
2	Braveheart	3404.0	7.7	1995	197	3.885388
43	The City of Lost Children	308.0	7.6	1995	902	3.677734
36	A Short Film About Killing	58.0	7.6	1989	10754	3.529266
3	The Big Sleep	244.0	7.6	1946	910	3.782963
17	Robin Williams: Weapons of Self Destruction	16.0	7.5	2009	26621	3.411855
34	Gabbeh	6.0	7.5	1996	43771	3.633399
1	Wings of Desire	256.0	7.5	1987	144	3.962263
40	Delicatessen	320.0	7.4	1991	892	4.008095
25	The Terminator	4208.0	7.4	1984	218	3.603920
4	Buena Vista Social Club	76.0	7.3	1999	11779	3.670132
13	Son of the Bride	37.0	7.3	2001	19460	3.581265

There were three search functions:

- Text Search (improved_recommendations)
- User Text Search (Hybrid)
- User Search

User Interface

The user interface was implemented using the Streamlit library. It consisted of a title section, a sidebar section for the project's general info, and two functional pages that could be switched using the first select box.



The screenshot shows a dark-themed web interface. At the top, the text "choose your desired recommendation method" is displayed. Below it is a dropdown menu with a red border. The menu is currently open, showing two options: "By User" and "By Text Search". The "By User" option is highlighted with a dark background. A small downward arrow is visible on the right side of the dropdown box.

The “By User” option indicates that the recommendation should be given according to the user’s favorite movies and ratings. Which could also use an optional movie pick for further precision.

In the user selection section, you can choose a range size for your possible user choices. Then you can pick one.

As mentioned earlier, there is an optional field for bonding the recommendation to a specific movie.

The field performs a full-text search on movies’ titles and genres.



Provide Your Desired Input

choose a range for possible choices

- ☐ small
- ☒ medium
- ☐ large
- ☐ huge

Please select the desired number of possible choices from the dataset



Pick a User

user_31

Enter a partial input for full-text search (optional)

god

submit

By hitting the recommendation button, if a text input is not mentioned, you'll observe a table of 10 recommended movies. Also, with the text search, you should pick your desired movie from the returned list; the same results will follow.

please select a movie

The Godfather

Continue

Done - Table Display may take a while

	title	vote_count	vote_average	year	id	est
284	The Shawshank Redemption	8,358.0000	8.5000	1994	278	4.4855
994	The Godfather: Part II	3,418.0000	8.3000	1974	240	4.3527
2998	The Conversation	377.0000	7.5000	1974	592	4.1694
986	GoodFellas	3,211.0000	8.2000	1990	769	4.1084
981	Apocalypse Now	2,112.0000	8.0000	1979	28	4.0781
2808	Midnight Express	309.0000	7.6000	1978	11327	3.8416
1346	The Rainmaker	239.0000	6.7000	1997	11975	3.7290
1765	The Paradine Case	42.0000	6.3000	1947	31667	3.6710
2742	...And Justice for All	118.0000	7.1000	1979	17443	3.6491
146	Feast of July	0.0000	0.0000	1995	259209	3.6068

The “By Text Search” option functions pretty much the same as the text field of the previous page, but the difference is that it’s not depended on the user, and therefore performs a general recommendation.

choose your desired recommendation method

By Text Search

Provide Your Desired Input

Enter a partial input for full-text search

god

Let The Magic Begin!

Recommend Me Some Movies

You may see that the results are not the same as the user-based ones.

please select a movie

The Godfather

Continue

Done - Table Display may take a while

	title	vote_count	vote_average	year	wr
284	The Shawshank Redemption	8358	8	1994	7.9336
994	The Godfather: Part II	3418	8	1974	7.8454
986	GoodFellas	3211	8	1990	7.8362
981	Apocalypse Now	2112	8	1979	7.7616
1602	The Godfather: Part III	1589	7	1990	6.8541
1100	Dracula	1087	7	1992	6.8013
2998	The Conversation	377	7	1974	6.5928
2808	Midnight Express	309	7	1978	6.5473
7464	The Bad Lieutenant: Port of Call - New Orleans	331	6	2009	6.0380
642	Jack	340	5	1996	5.5057

The searchings were performed by the search functions implemented on the model.py file, which was copied from the actual model that worked on the dataset.

Data files in the data folder, which were used for our functionality, were the processed data by the actual model (available on the Colab notebook) that went through cleaning, encoding, and other processing.

Two other files were also implemented:

- A collaborative API with the model (api.py)
- A text format helper for the UI's texts (textFormatter.py)

Deployment

The model was deployed on the Hugging face space and the Colab notebook.

The mentioned deployments can be accessed using the links below:

[Hugging Face](#)

[Colab](#)