

Movie Recommendation system in Python

A course project report

By

Aryan Sinha[RA2011003010066]

Gajulapalli Naga Vyshnavi [RA2011003010049]

Shruthi Kannan [RA2011003010037]

Under the guidance of

Mr. Ghuntupalli Manoj Kumar N

(Assistant Professor, Department of Computing Technologies)

In partial fulfillment for the course of

18CSC305J - Artificial Intelligence

in the department of Computing Technologies,

School of Computing,

Faculty of Engineering and Technology,

For the degree of

Bachelor of Technology in Computer Science and Engineering



SRM

INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu Taluk, Kanchipuram District, Tamil Nadu, India - 603203

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)



BONAFIDE CERTIFICATE

Certified that this project report titled “Movie Recommendation system in Python” is the bonafide work of “Aryan Sinha”, “Gajulapalli Naga Vyshnavi”, ”Shruthi Kannan” who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr Ghuntupalli Manoj Kumar N

Assistant Professor,

Department of Computing Technologies,
SRM Institute of Science and Technology

TABLE OF CONTENTS

1. AIM
2. ABSTRACT
3. INTRODUCTION
4. IMPLEMENTATION
5. EXPLANATION
6. CONCLUSION
7. RESULT
8. REFERENCES

AIM

The aim of our project is to build a customized movie recommendation system where in similar movies will be recommended to users based on the ratings of the movies watched by other users.

ABSTRACT

This project aims to implement a movie recommendation system using collaborative filtering in python and perform exploratory data analysis on the datasets used. In order to gain valuable insights to make recommendations for users based on ratings of other users who have watched movies with similar ratings to recommend further movies to watch data frames, collaborative filtering techniques will be applied.

This will be done by working and pre-processing a dataset of user-movie ratings which will be stored in a data frame. The data frame is useful in order to manipulate, filter, sort the data by user movie id or ratings which will then be visualized by visualization libraries like seaborn in order to provide useful insights into the similarities and outliers in the data.

INTRODUCTION

Collaborative filtering is a commonly used technique in movie recommendation systems that aims to recommend movies to users based on their similarities to other users with similar tastes. The idea is that if two users have similar movie preferences in the past, then they are likely to have similar preferences in the future as well.

In a movie recommendation system, collaborative filtering can be implemented in several ways, such as user-based collaborative filtering and item-based collaborative filtering. In user-based collaborative filtering, the system recommends movies to a user based on the preferences of other similar users. In item-based collaborative filtering, the system recommends movies that are similar to the movies the user has already watched and liked.

Python is a popular language for implementing movie recommendation systems using collaborative filtering. The process involves collecting user data and movie data, preparing the data for analysis, and applying the collaborative filtering algorithm to make recommendations. Popular Python libraries used for implementing collaborative filtering include Pandas, NumPy.

Exploratory Data Analysis (EDA) is a crucial step in the movie recommendation system project in Python. EDA is a process of analysing and visualizing the data to understand its characteristics and extract meaningful insights from it.

In a movie recommendation system project, EDA involves analysing and understanding the dataset, which contains information about the users, movies, and their ratings. EDA can help to identify patterns, trends, and relationships within the data that can be used to improve the recommendation system's accuracy and effectiveness.

Some common techniques used in EDA for movie recommendation systems include data visualization techniques such as histograms, scatter plots, and heatmaps. These visualizations can help to identify outliers, correlations, and other patterns in the data that may not be immediately apparent through summary statistics alone.

IMPLEMENTATION:

In this project we aim on building a movie recommendation system using collaborative filtering in Python. The system will use datasets of user-movie ratings to identify patterns of similarity between users and movies and make recommendations based on these patterns. The datasets will be pre-processed and cleaned, followed by visualizing the distribution of the data using histograms. Histograms are a powerful tool in EDA as they allow us to explore the shape, spread, and outliers in the data.

We have used Seaborn library to implement visualizations, like joint plots and histograms, which will help in exploring relationships between variables and figure out outliers in the data. Seaborn provides a range of customizable visualizations that can enhance the analysis of the dataset.

The objective of this method is to gain insights, understand trends and outliers in the dataset using exploratory data analysis. The insights gained from the analysis can be used to make data-driven decisions and guide further analysis.


Data frames will be used to store information about movies, such as title, genre, year of release, actors, directors, and ratings. The data frame can be created by reading in a CSV or Excel file, or by scraping data from a website or API, and are used in order to perform operations such as filtering, sorting, and grouping to extract useful information about the movies. For example, the data frame can be filtered to only include movies from a certain genre or year, sort the data frame based on ratings or popularity, or group the data frame by director or actor.


After the data has been cleaned, processed the data frame, you can use it to build a recommendation engine. The approach we have used is collaborative filtering, which involves finding similarities between users and recommending movies that similar users have enjoyed. With the help of ratings data in the data frame a matrix can be built which will represent the ratings of various users for each movie and then using correlation similarity between movies based on user ratings can be found. In particular, we can use the Pearson correlation coefficient, which measures the linear correlation between two variables, to compute the similarity between two movies based on the ratings given by users. If two movies have a high positive correlation coefficient, it means that they tend to be rated similarly by users. On the other hand, if two movies have a high negative correlation coefficient, it means that they tend to be rated differently by users.

To use correlation in a movie recommender system, we can start by creating a user-movie rating matrix, where each row represents a user, and each column represents a movie. The values in the matrix represent the ratings given by users to movies. We can then compute the correlation between each pair of movies based on the ratings given by users. We can use the resulting correlation matrix to find movies that are like a given movie.

For example, suppose a user has rated the movie "The Dark Knight" highly. We can use the correlation matrix to find other movies that are highly correlated with "The Dark Knight", indicating that users who liked "The Dark Knight" also tended to like those movies. We can then recommend those highly correlated movies to the user.

CODE AND OUTPUTS:

jupyter movie recommendation system Last Checkpoint: 11 minutes ago (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) 

In [1]: `import numpy as np
import pandas as pd`

In [2]: `column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=column_names)`

In [3]: `df.head()`


Out[3]:


	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

In [4]: `movie_titles = pd.read_csv("Movie_Id_Titles")
movie_titles.head()`

Out[4]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

jupyter movie recommendation system Last Checkpoint: 13 minutes ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) 

In [5]: `df = pd.merge(df, movie_titles, on='item_id')
df.head()`

Out[5]:

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

In [6]: `import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
%matplotlib inline`

In [7]: `df.groupby('title')['rating'].mean().sort_values(ascending=False).head()`

Out[7]:

title	rating
They Made Me a Criminal (1939)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Saint of Fort Washington, The (1993)	5.0
Someone Else's America (1995)	5.0
Star Kid (1997)	5.0

Name: rating, dtype: float64

In [8]: `df.groupby('title')['rating'].count().sort_values(ascending=False).head()`

Out[8]:

title	rating
Star Wars (1977)	584
Contact (1997)	509
Fargo (1996)	508
Return of the Jedi (1983)	507
Liar Liar (1997)	485

Name: rating, dtype: int64

File Edit View Insert Cell Kernel Widgets Help

Run Code

Name: rating, dtype: int64

```
In [9]: ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()
```

```
Out[9]:
```

	rating
title	
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

```
In [10]: ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

```
Out[10]:
```

	rating	num of ratings
title		
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

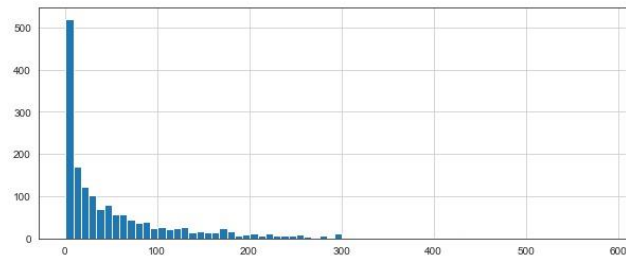
File Edit View Insert Cell Kernel Widgets Help

Run Code

187 (1997) 3.024390 41

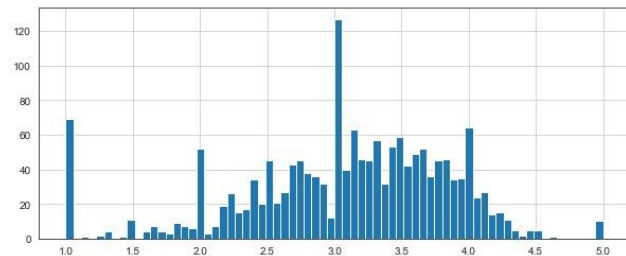
```
In [11]: plt.figure(figsize=(10,4))
ratings['num of ratings'].hist(bins=70)
```

```
Out[11]: <AxesSubplot:>
```



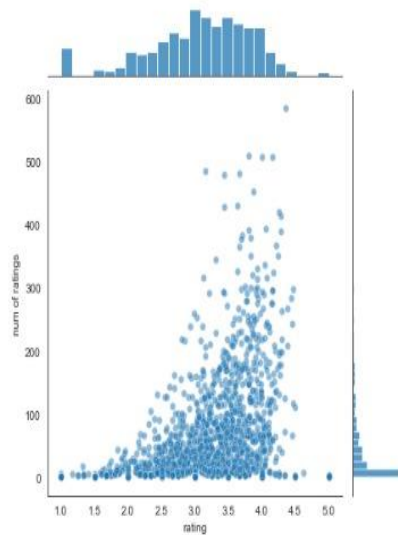
```
In [12]: plt.figure(figsize=(10,4))
ratings['rating'].hist(bins=70)
```

```
Out[12]: <AxesSubplot:>
```



```
In [13]: sns.jointplot(x='rating',y='num of ratings',data=ratings,alpha=0.5)
```

```
Out[13]: <seaborn.axisgrid.JointGrid at 0x27f330ef970>
```



```
In [14]: moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

```
Out[14]:
```

	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Y Poisc Hand The (
user_id																	
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	NaN	NaN	NaN	5.0	3.0	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 1664 columns

```
In [15]: ratings.sort_values('num of ratings',ascending=False).head(10)
```

```
Out[15]:
```

	rating	num of ratings
title		
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429


```
In [16]: ratings.head()
```

```
Out[16]:
```

	rating	num of ratings
title		
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

```
In [17]: starwars_user_ratings = moviemat['Star Wars (1977)']
liarliar_user_ratings = moviemat['Liar Liar (1997)']
starwars_user_ratings.head()
```

```
Out[17]: user_id
0      5.0
1      5.0
2      5.0
3      NaN
4      5.0
Name: Star Wars (1977), dtype: float64
```

jupyter movie recommendation system Last Checkpoint: 20 minutes ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

```
In [18]: similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)

C:\Users\VAISHNAVI\anaconda3\New folder\lib\site-packages\numpy\lib\function_base.py:2634: RuntimeWarning: Degrees of freedom <
= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\Users\VAISHNAVI\anaconda3\New folder\lib\site-packages\numpy\lib\function_base.py:2493: RuntimeWarning: divide by zero enco
ntered in true_divide
  c *= np.true_divide(1, fact)
```

```
In [19]: corr_starwars = pd.DataFrame(similar_to_starwars, columns=['Correlation'])
corr_starwars.dropna(inplace=True)
corr_starwars.head()
```


Out[19]:

	Correlation
title	
Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

```
In [20]: corr_starwars.sort_values('Correlation', ascending=False).head(10)
```

Out[20]:

	Correlation
title	
Commandments (1997)	1.0
Cosi (1996)	1.0
No Escape (1994)	1.0
Stripes (1981)	1.0
Man of the Year (1995)	1.0
Hollow Reed (1996)	1.0
Beans of Egypt, Maine, The (1994)	1.0
Good Man in Africa, A (1994)	1.0
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991)	1.0
Outlaw, The (1943)	1.0

jupyter movie recommendation system Last Checkpoint: 21 minutes ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

```
In [21]: corr_starwars = corr_starwars.join(ratings['num of ratings'])
corr_starwars.head()
```

Out[21]:

	Correlation	num of ratings
title		
Til There Was You (1997)	0.872872	9
1-900 (1994)	-0.645497	5
101 Dalmatians (1996)	0.211132	109
12 Angry Men (1957)	0.184289	125
187 (1997)	0.027398	41

```
In [22]: corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation', ascending=False).head()
```

Out[22]:

	Correlation	num of ratings
title		
Star Wars (1977)	1.000000	584
Empire Strikes Back, The (1980)	0.748353	368
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

```
In [24]: corr_liarliar = pd.DataFrame(similar_to_liarliar, columns=['Correlation'])
corr_liarliar.dropna(inplace=True)
corr_liarliar = corr_liarliar.join(ratings['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation', ascending=False).head
```

Out[24]:

	Correlation	num of ratings
title		
Liar Liar (1997)	1.000000	
Batman Forever (1995)	0.516968	
Mask, The (1994)	0.484650	
Down Periscope (1996)	0.472681	
Con Air (1997)	0.469828	
...	...	
Hoop Dreams (1994)	-0.184503	
Ed Wood (1994)	-0.199481	
Dr. Strangelove or: How I Learned to Stop Worry...	-0.238092	
Welcome to the Dollhouse (1995)	-0.254231	
Raging Bull (1980)	-0.308129	

EXPLANATION:

The codes implement a movie recommendation system project using Python, utilizing the Pandas and NumPy libraries for data manipulation and analysis, and the Seaborn and Matplotlib libraries for data visualization.

First, the dataset of user ratings for movies and a dataset containing the titles of the movies is read. It then merges the two datasets using the `item_id` column to create a new dataset that includes both the ratings and titles of the movies. Then the code performs exploratory data analysis (EDA) on the dataset. It groups the movies by their title and calculates the average rating and number of ratings for each movie, creating a new data frame with this information. The code then visualizes the distribution of the number of ratings and ratings using histograms and a joint plot.

The code then creates a pivot table of the user ratings, with the rows representing the `user_id`, columns representing the movie titles, and the values representing the ratings. It uses this table to calculate the correlation between the user ratings for specific movies and other movies in the dataset. Finally, the code selects two movies, "Star Wars (1977)" and "Liar Liar (1997)," and calculates the correlation between their ratings and the ratings of other movies in the dataset. It filters the results to only include movies with more than 100 ratings and displays the top results in descending order of correlation.

Overall, the code demonstrates how to perform a basic movie recommendation system using collaborative filtering and EDA techniques in Python.

RESULT:

We successfully implemented a movie recommendation system using collaborative filtering for the python language.

REFERENCES:

- <https://www.geeksforgeeks.org/user-based-collaborative-filtering/>
- <https://medium.com/analytics-vidhya/simple-movie-recommender-system-with-correlation-coefficient-with-python-e6cb31dae01e>
- <https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/>