MOVIE RECOMMENDATION SYSTEM

MINI PROJECT

(FUNDAMENTALS OF MACHINE LEARNING)

Submitted by:

SANYAM JAIN (9919103094)

विद्या तत्व ज्योतिसमः

Department of CSE/IT
Jaypee Institute of Information Technology University, Noida

December 2021

ABSTRACT

We all watch movies, in our leisure time, and spend a lot of time to find movies according to our taste. Using concepts of machine learning, this project helps user to find movies similar to the movies they enjoyed.

This project takes input of a movie you liked, and the number of movies to want request, and gives movies with their ratings, in descending order of correlation. Using basic concepts of machine learning, that is, Correlation. This project follows item-based filtering.

INTRODUCTION

Problem statement: To find similar movies as the given input, in the given dataset.

Objectives:

- Predict movies, and print them with their ratings, in descending order of correlation.
- The prediction should be accurate and should not be too vague.

BACKGROUND STUDY

Anaconda: is a distribution of the Python, The distribution includes data-science packages suitable for Windows, Linux, and macOS.

Jupyter Notebook: is an excellent open-source web application that allows you to create and share documents that contain live code, equations, visualizations and used for data cleaning and transformation, numerical simulation, statistical modelling, data visualization and machine learning.

Python Libraries used: Pandas, Matplotlib.

ALGORITHM

corrwith(): is an inbuilt function, it computes pairwise correlation. Pairwise correlation is computed between rows or columns of DataFrame with rows or columns of Series or DataFrame. DataFrames are first aligned along both axes before computing the correlations.

Method of correlation: Pearson correlation coefficient also known as Pearson's r, the Pearson product-moment correlation coefficient (PPMCC), the bivariate correlation, as the correlation coefficient is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalized measurement of the covariance, such that the result always has a value between −1 and 1(least similar: -1; most similar: 1).

IMPLEMENTATION

Jupyter Notebook:

In [1]:
```
import pandas as pd
import warnings
import matplotlib.pyplot as plt
```

In [2]:
```
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv("ml-latest-small/movies.csv")
df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```
item = pd.read_csv("ml-latest-small/ratings.csv")
item.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [5]:

```
df = pd.merge(df, item, on ='movieId')
df
```

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1	4.0	964982703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	847434962
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	4.5	1106635946
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.5	1510577970
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	17	4.5	1305696483
...	...	...	...	...	...	...
100831	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy	184	4.0	1537109082
100832	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy	184	3.5	1537109545
100833	193585	Flint (2017)	Drama	184	3.5	1537109805
100834	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation	184	3.5	1537110021
100835	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy	331	4.0	1537157606

100836 rows × 6 columns

In [6]:

```
df.drop(columns =['timestamp', 'genres'], inplace = True)
```

In [7]:

```
ratings = pd.DataFrame(df.groupby('title').mean()['rating'])
ratings['freq'] = pd.DataFrame(df.groupby('title').count()['rating'])
ratings.head()
```

	rating	freq
title		
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

In [8]:

```
mat = df.pivot_table(index ='userId', columns = 'title', values = 'rating')
mat
```

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)
userId														
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
3	NaN	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
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
606	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
608	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
609	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
610	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.5	NaN	...	NaN	4.0	3.5

610 rows × 9719 columns

```
In [9]: def get_movie(movie_name, number_of_items):
            movie_user_rating = mat[movie_name]
            similar_item = mat.corrwith(movie_user_rating)
            corr_item = pd.DataFrame(similar_item, columns = ['Correlation'])
            corr_item.dropna(inplace = True)

            corr_item = corr_item.join(ratings['freq'])
            corr_item = corr_item.join(ratings['rating'])
            prediction = corr_item[corr_item['freq'] > 100].sort_values('Correlation', ascending = False)

            prediction = prediction[['rating', 'Correlation']]
            ans =  prediction[1:].head(number_of_items)
            return ans;
```

```
In [16]: print("Enter movie you liked: ")
         movie_name = input()
         print("Enter how many recommendations would you like: ")
         number_of_items = int(input())
         ans = get_movie(movie_name, number_of_items)
         print(ans)
```

```
Enter movie you liked:
Incredibles, The (2004)
Enter how many recommendations would you like:
3
                         rating  Correlation
title
Toy Story (1995)       3.920930     0.643301
Finding Nemo (2003)    3.960993     0.561018
Monsters, Inc. (2001)  3.871212     0.544516
```

EXPERIMENTAL RESULTS

DATASET:

- Dataset used in this project is taken from website of grouplens, the one that is mentioned for development purposes.
- Shape of movie dataset is: (9743, 3)
- Shape of rating dataset is: (100837, 4)

OUTPUT:

-

```
Enter movie you liked:
Incredibles, The (2004)
Enter how many recommendations would you like:
3
                           rating  Correlation
title
Toy Story (1995)         3.920930     0.643301
Finding Nemo (2003)      3.960993     0.561018
Monsters, Inc. (2001)    3.871212     0.544516
```

Here input is given as “Incredibles, The (2004)”, which is a Disney animated movie, and in output we get other Disney animated movies like Toy story.

-

```
Enter movie you liked:
Star Wars: Episode IV - A New Hope (1977)
Enter how many recommendations would you like:
2
                                                   rating  Correlation
title
Star Wars: Episode V - The Empire Strikes Back ...  4.215640      0.77797
Star Wars: Episode VI - Return of the Jedi (1983)   4.137755      0.73423
```

Here input is the first movie in Star Wars franchise, and the output are the sequels which are highly correlated.

-

```
Enter movie you liked:
Lord of the Rings: The Two Towers, The (2002)
Enter how many recommendations would you like:
2
                                                      rating  Correlation
title
Lord of the Rings: The Fellowship of the Ring, ...  4.106061     0.887301
Lord of the Rings: The Return of the King, The ...  4.118919     0.821503
```

Here input is the second movie in the Lord of rings franchise, and we get the output of the other 2 movies, the prequel and the sequel to the given input.

CONCLUSION

The code is working fine as per the above test cases. We are getting movies from the same franchise whatever may be the order. But the parameters are still less to get a greater number of accurate results. In order to achieve this, a bigger dataset has to be used.