REPORT FOR RECOMMENDER SYSTEMS

As a project work for Course

PYTHON PROGRAMMING (INT 213)

Name of student 1 : Ramchandra Choudhary

Registration Number : 12016775

Name of student 2 : Abhishek Mishra

Registration Number : 12016771

Program : BTech. CSE

Semester : 3rd

School : School of Computer Science Engineering

Name of the University : **LOVELY PROFESSIONAL UNIVERSITY**

Date of Submission **: 20th November 2021**

Lovely Professional University , Jalandhar , Punjab , India

RECOMMENDER SYSTEMS

20th November 2021

ABSTRACT:-

Recommendation systems attempt to predict the preference or rating that a user would give to an item. Knowledge discovery techniques can be applied to the problem of making personalized recommendations about items or information during a user's visit to a website. Collaborative Filtering algorithms give recommendations to a user based on the ratings of other users in the system.

Traditional collaborative filtering algorithms face issues such as scalability, sparsity, and cold start. In the proposed framework, prediction using item based collaborative filtering is combined with prediction using demographics-based user clusters in an adaptive weighted scheme. The proposed solution will be scalable while addressing user cold start.

ACKNOWLEDGEMENT:-

I would like to thank my mentor –**Ankita Wadhawan** for his advice and inputs on this project. Many thanks to my friends and seniors as well, who spent countless hours to listen and provide feedbacks.

INTRODUCTION:

Context

This project has been done as part of my course for the CSE at Lovely Professional University Supervised by **Ankita Wadhawan**, I have one months to fulfil the requirements to succeed the module.

Motivations

Being extremely interested in everything having a relation with the Python , the group project was a great occasion to give us the time to learn and confirm our interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use this system in Medicine , Media , Products and almost in everywhere. That's why I decided to do this project.

Idea

A recommendation system targets the specific interests of consumers. Phrased differently, it prevents a premature end of a customer journey due to overload or frustration. That is why recommendation systems are crucial for enterprises with a broad range of goods or services .It is also essential to keep in mind the customers' usage behaviour. Whereas some pages encourage extensive browsing, others are designed to identify the optimal product as quickly as possible

Theory about Recommendation system:-

Recommender systems are among the most popular applications of data science today. They are used to predict the "rating" or "preference" that a user would give to an item. Almost every major tech company has applied them in some form. Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow.

What's more, for some companies like Netflix, Amazon Prime, Hulu, and Hotstar, the business model and its success revolves around the potency of their recommendations. Netflix even offered a million dollars in 2009 to anyone who could improve its system by 10%.

Types of recommender system:-

- **Simple recommenders:** offer generalized recommendations to every user, based on movie popularity and/or genre. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. An example could be IMDB Top 250.

- **Content-based recommenders:** suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person likes a particular item, he or she will also like an item that is similar to it. And to recommend that, it will make use of the user's past item metadata. A good example could be YouTube, where based on your history, it suggests you new videos that you could potentially watch.

- **Collaborative filtering engines:** these systems are widely used, and they try to predict the rating or preference that a user would give an item-based on past ratings and preferences of other users. Collaborative filters do not require item metadata like its content-based counterparts.

In this project I am creating a

SIMPLE RECOMMENDER SYSTEM :-

Simple recommenders are basic systems that recommend the top items based on a certain metric or score. In this section, you will build a simplified clone of IMDB Top 250 Movies using metadata collected from IMDB.

This dataset consists of the following files:

1) movies_metadata.csv : This file contains all the data about the movies like name, genre, release date , votes , rating etc. Its having data of almost 45k movies.

Since you are trying to build a clone of IMDB's Top 250, let's use its weighted rating formula as a metric/score. Mathematically, it is represented as follows:

$$Weighted Rating(\mathbf{WR}) = \left(\frac{\mathbf{v}}{\mathbf{v}+\mathbf{m}} \cdot \mathbf{R}\right) + \left(\frac{\mathbf{m}}{\mathbf{v}+\mathbf{m}} \cdot \mathbf{C}\right)$$

In the above equation,

- v is the number of votes for the movie
- m is the minimum 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.

TEAM MEMBERS (contributions):-

Ramchandra Choudhary

Contributions:-

- Theory and Basics
- Coding (joined)
- Pandas (machine learning application)
- Reports(joined)

Abhishek Mishra

Contributions:-

- Coding(joined)
- Pandas
- Reports
- Datasets

LIBRARIES:-

Pandas:-

Pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series.

To load your dataset, you would be using the **pandas** DataFrame library. library is mainly used for data manipulation and analysis. It represents your data in a row-column format . Pandas library is backed by the NumPy array for the implementation of pandas data object. Pandas offer off the shelf data structures and operations for manipulation of numerical tables , time-series, imagery, and natural language processing datasets.

Screenshots:-

MOVIE RECOMMENDATION SYSTEM

```
[1] # IMPORT PANDAS
    import pandas as pd
```

```
[2] #  LOADING METADATA OF MOVIES
    metadata = pd.read_csv('movies_metadata.csv', low_memory=False)
```

```
[3] # Print the first three rows
    metadata.head(5)
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	po
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	2
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	1

```
[4] # Calculate mean of vote average column
    C = metadata['vote_average'].mean()
    print(C)
```

```
5.618207215133889
```

```
[5] # Calculate the minimum number of votes required to be in the chart, m
    m = metadata['vote_count'].quantile(0.90)
    print(m)
```

```
160.0
```

```
[6] # Filter out all qualified movies into a new DataFrame          See , we are dealing with around 4.5k movies
    q_movies = metadata.copy().loc[metadata['vote_count'] >= m]
    q_movies.shape
```

```
(4555, 24)
```

```
    # This is the original shape of the metadat_csv file . It has around 45k movies
    metadata.shape
```

```
(45466, 24)
```

```
[8] # Function that computes the weighted rating of each movie

    def weighted_rating(x, m=m, C=C):
        v = x['vote_count']
        R = x['vote_average']
```

```
[8] # Function that computes the weighted rating of each movie

    def weighted_rating(x, m=m, C=C):
        v = x['vote_count']
        R = x['vote_average']
        # Calculation based on the IMDB formula
        return (v/(v+m) * R) + (m/(m+v) * C)
```

```
[9] # Define a new feature 'score' and calculate its value with `weighted_rating()`
    q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

```
[10] # Sort movies in ascending order based on score calculated above
     q_movies = q_movies.sort_values('score', ascending=False)
```

```
# Print the top recommended movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(20)
```

	title	vote_count	vote_average	score
314	The Shawshank Redemption	8358.0	8.5	8.445869
834	The Godfather	6024.0	8.5	8.425439
10309	Dilwale Dulhania Le Jayenge	661.0	9.1	8.421453
12481	The Dark Knight	12269.0	8.3	8.265477
2843	Fight Club	9678.0	8.3	8.256385

REFRENCES:-

To conduct this project the following tools have been used

- Google colab
- Pandas (Library) : http://pandas.pydata.org/

- https://www.datacamp.com/community/tutorials/recommender-systems-python
- https://www.geeksforgeeks.org/recommendation-system-in-python/
- https://realpython.com/build-recommendation-engine-collaborative-filtering/
- https://www.google.com